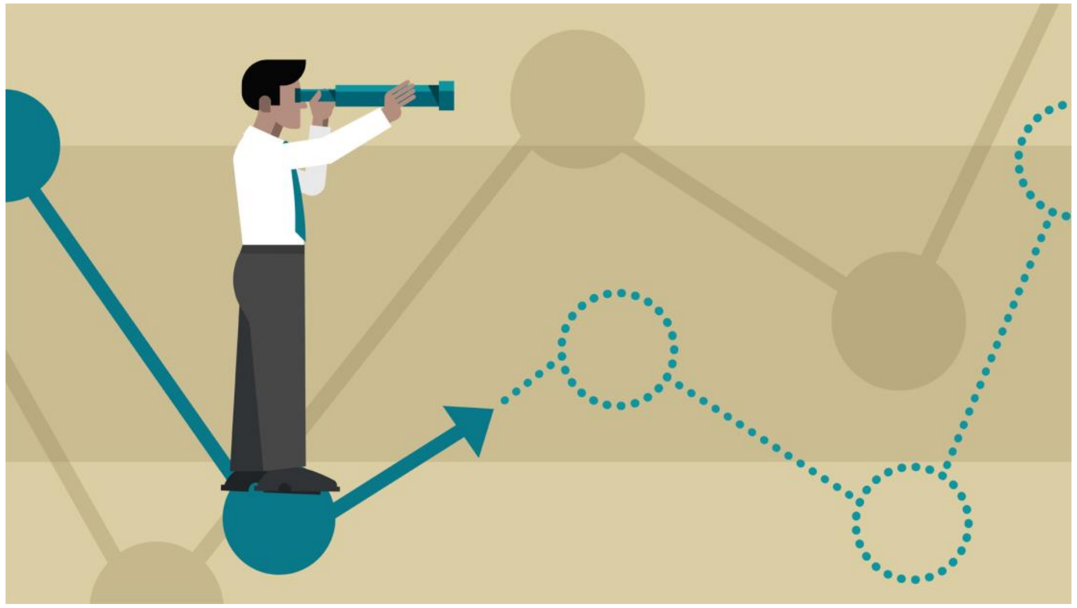
Forecasting through Support Vector Machines

Business Intelligence per i Servizi Finanziari 2023-2024

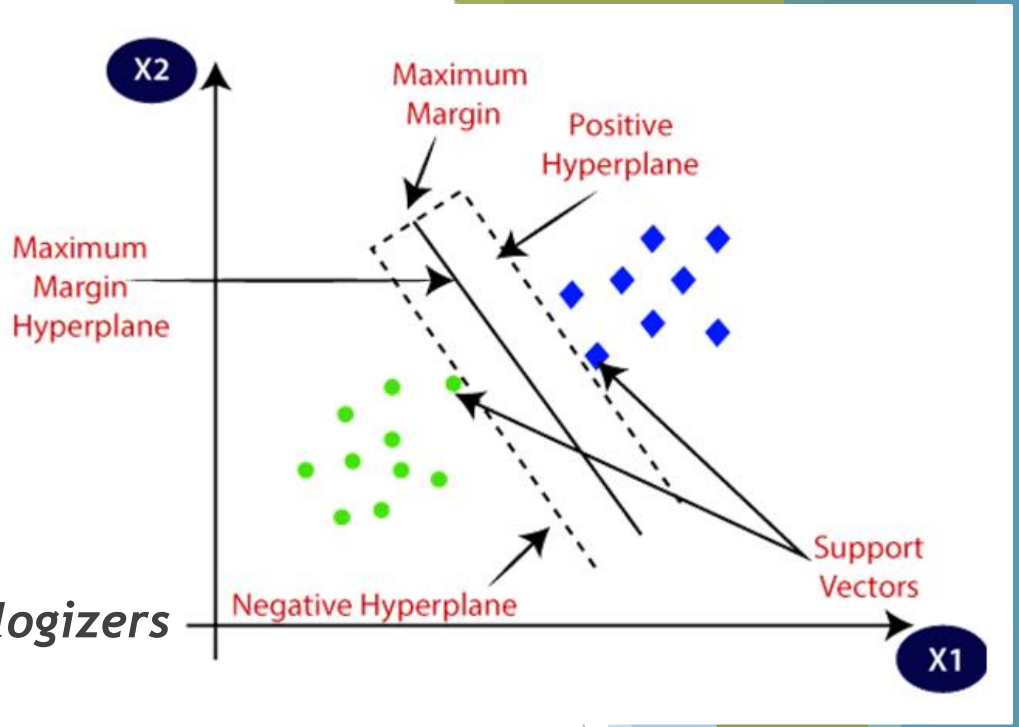
Antonio Candelieri

Forecasting

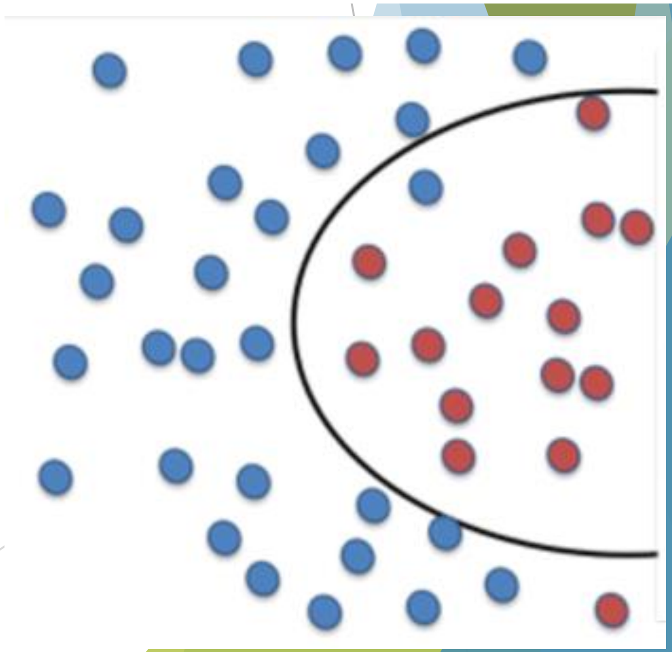
- In the analysis of financial time series we are interested in designing models for **predicting future values.**



Support Vector Machine (SVM)



- Widely used for classification learning problems
- According to its learning paradigm, SVM belongs to the ***Analogizers***
- SVM classifier: maximum separation hyperplane
- Support vectors
- Linear and non-linear classification



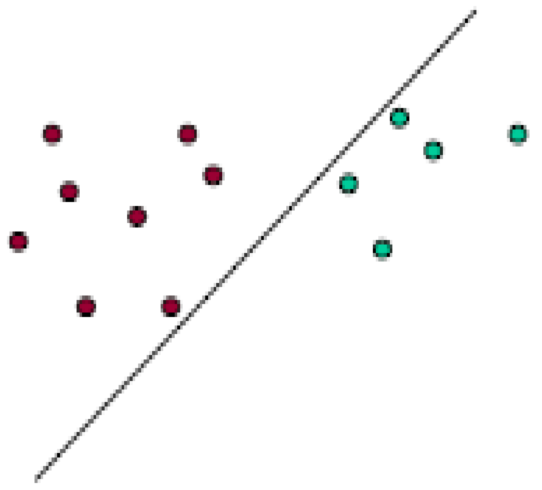
Support Vector Machine classification

- Basic idea of support vector machines
  - **Optimal** hyperplane for **linearly separable** patterns
  - **Extend to** patterns that are **not linearly separable** by transformations of original data to map into new space → *kernel function*

Unique properties of SVM

- Are explicitly based on a theoretical model of learning
- Come with theoretical guarantees about their performance
- Are not affected by local minima
- Do not suffer from the curse of dimensionality

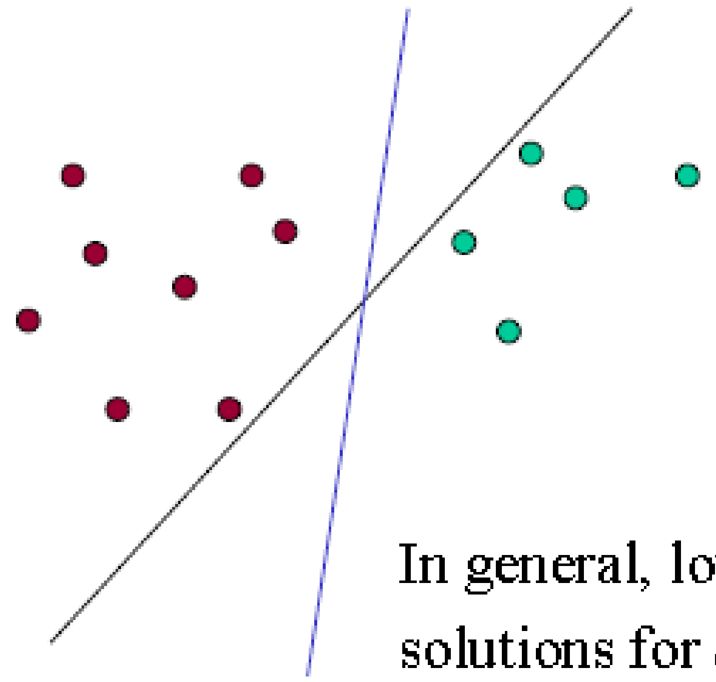
Linear Separability



Find a,b,c, such that

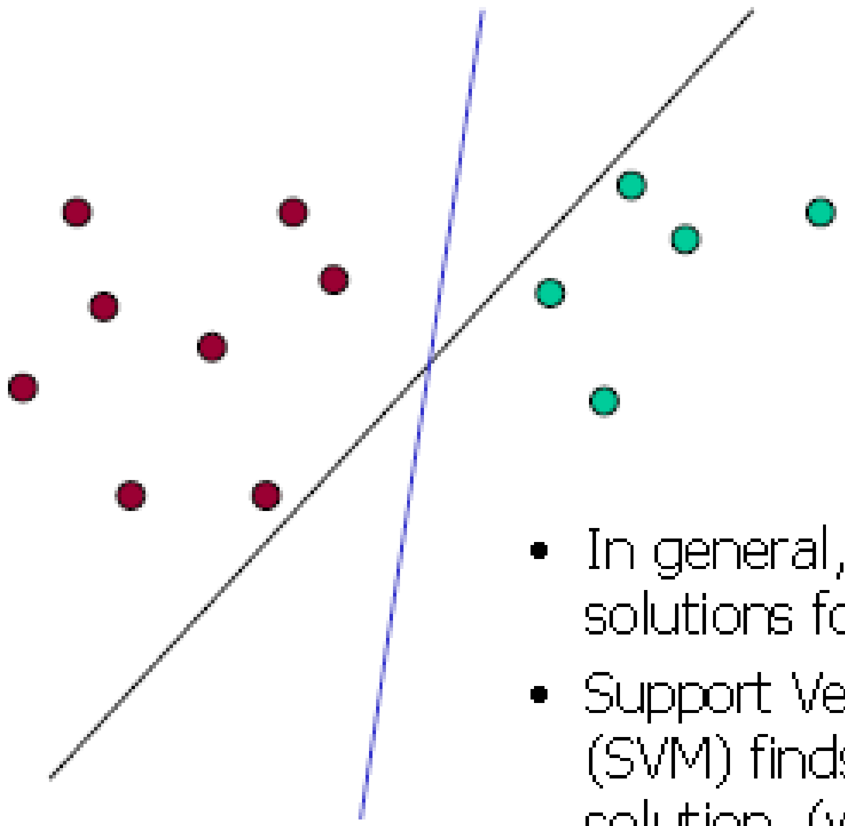
$ax + by \geq c$ for red points

$ax + by \leq c$ for green points.



In general, lots of possible solutions for a,b,c.

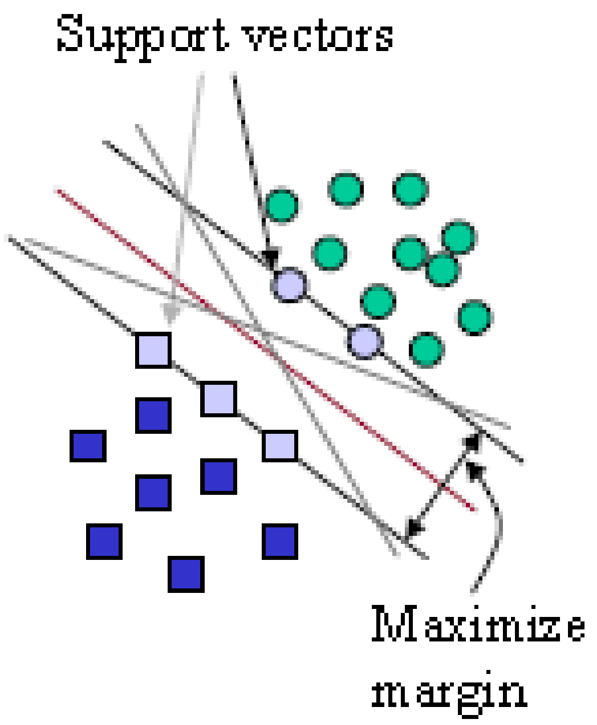
Separating hyperplane



- In general, lots of possible solutions for *a,b,c*.
- Support Vector Machine (SVM) finds an <u>optimal</u> solution. (wrt what cost?)

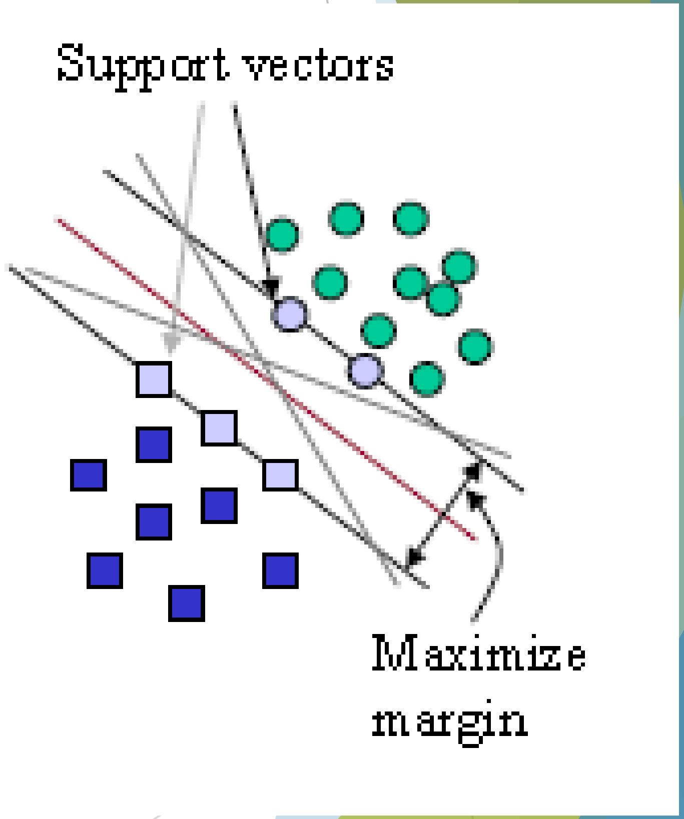
SVM classification learning

- SVMs maximize the *margin* around the separating hyperplane.
- The decision function is fully specified by a subset of training samples, *the support vectors*.
- *Quadratic programming* problem



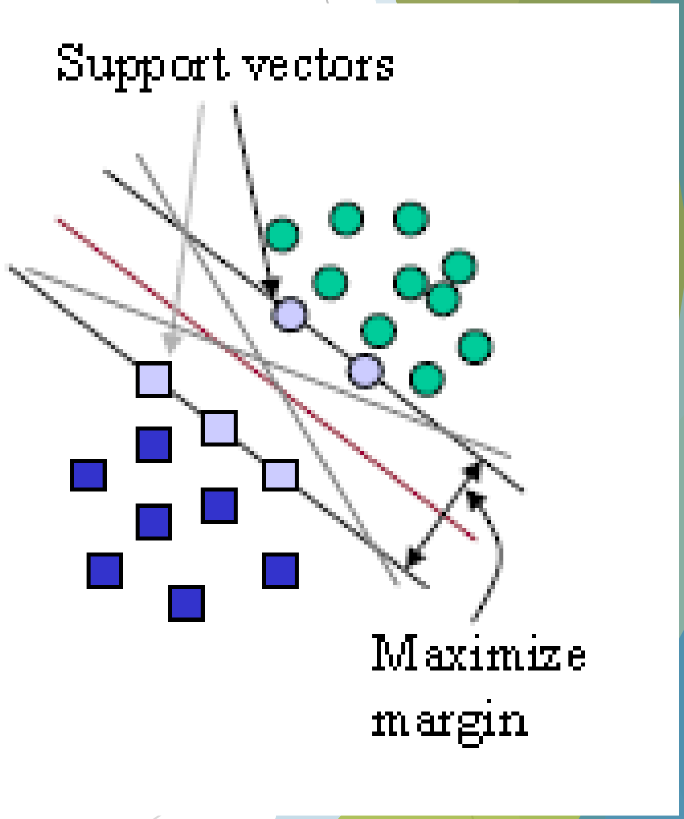
Support Vectors

- **Support vectors** are the data points that lie closest to the decision surface
- They are the most difficult to classify
- They have direct bearing on the optimum location of the decision surface



Support Vectors

- Support vectors are the elements of the training set that would change the position of the dividing hyperplane if removed
- Support vectors are the critical elements of the training set
- The problem of finding the optimal hyperplane is an optimization problem and can be solved by optimization techniques (use Lagrange multipliers to get into a form that can be solved analytically).



Separating hyperplane and Support Vectors

Define the hyperplane H such that:

$x_i \bullet \mathbf{w} + b \geq +1$ when $y_i = +1$

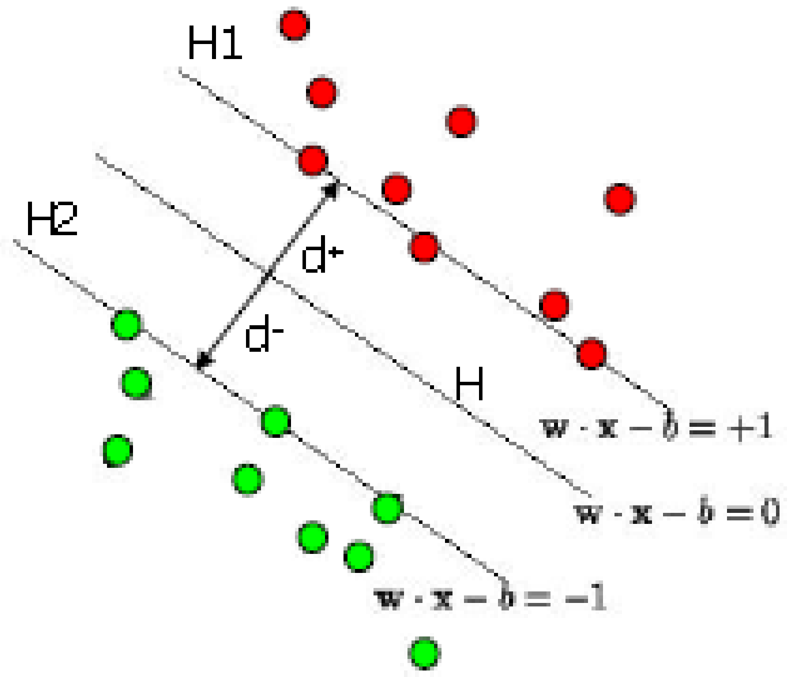
$x_i \bullet \mathbf{w} + b \leq -1$ when $y_i = -1$

H1 and H2 are the planes:

H1: $\mathbf{x}_i \bullet \mathbf{w} + b = +1$

H2: $\mathbf{x}_i \bullet \mathbf{w} + b = -1$

The points on the planes H1 and H2 are the Support Vectors


H1
H2
d+
d-
H
$\mathbf{w} \cdot \mathbf{x} - b = +1$
$\mathbf{w} \cdot \mathbf{x} - b = 0$
$\mathbf{w} \cdot \mathbf{x} - b = -1$

d+ = the shortest distance to the closest positive point

d- = the shortest distance to the closest negative point

The margin of a separating hyperplane is $d^+ + d^-$.

SVM classification: Hard-margin formulation

Let's assume we can separate the data perfectly. Then we can optimize the following:

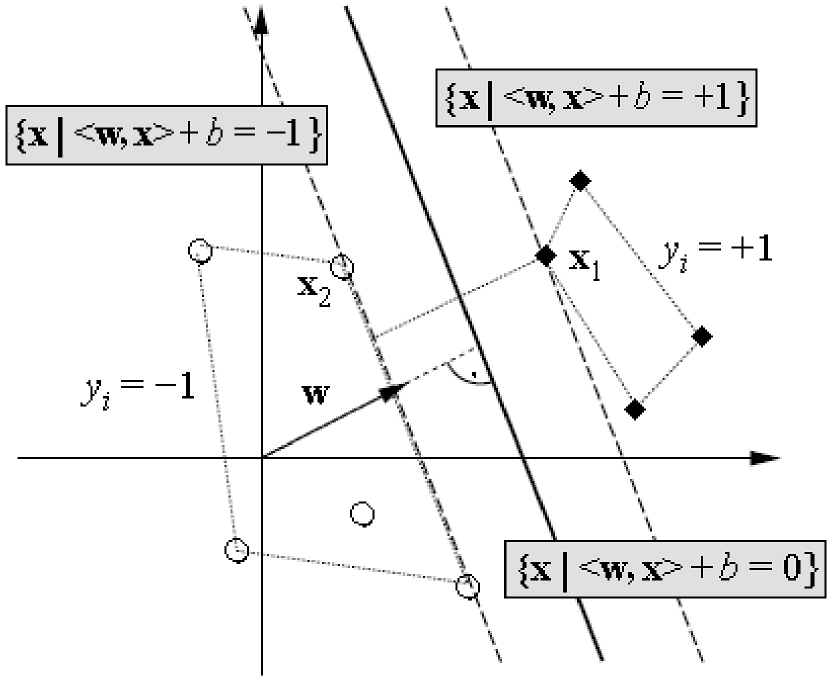
Minimize $||w||$, subject to:

$$(w \cdot x_i + b) \geq 1, \text{ if } \; y_i = 1$$
$$(w \cdot x_i + b) \leq -1, \text{ if } \; y_i = -1$$

The last two constraints can be compacted to:

$$y_i(w \cdot x_i + b) \geq 1$$

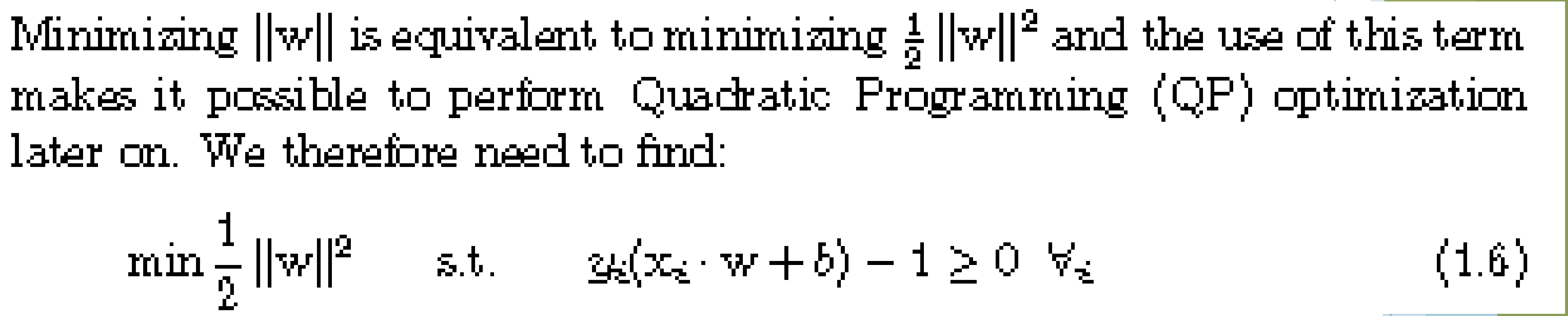
This is a quadratic program.


$\{\mathbf{x} \mid \langle\mathbf{w}, \mathbf{x}\rangle + b = -1\}$
$\{\mathbf{x} \mid \langle\mathbf{w}, \mathbf{x}\rangle + b = +1\}$
$\mathbf{x}_1$
$y_i = +1$
$\mathbf{x}_2$
$y_i = -1$
$\mathbf{w}$
$\{\mathbf{x} \mid \langle\mathbf{w}, \mathbf{x}\rangle + b = 0\}$

Note:

$$\langle\mathbf{w}, \mathbf{x}_1\rangle + b = +1$$
$$\langle\mathbf{w}, \mathbf{x}_2\rangle + b = -1$$
$$\Rightarrow \quad \langle\mathbf{w}, (\mathbf{x}_1 - \mathbf{x}_2)\rangle = 2$$
$$\Rightarrow \quad \left\langle \frac{\mathbf{w}}{||\mathbf{w}||}, (\mathbf{x}_1 - \mathbf{x}_2) \right\rangle = \frac{2}{||\mathbf{w}||}$$

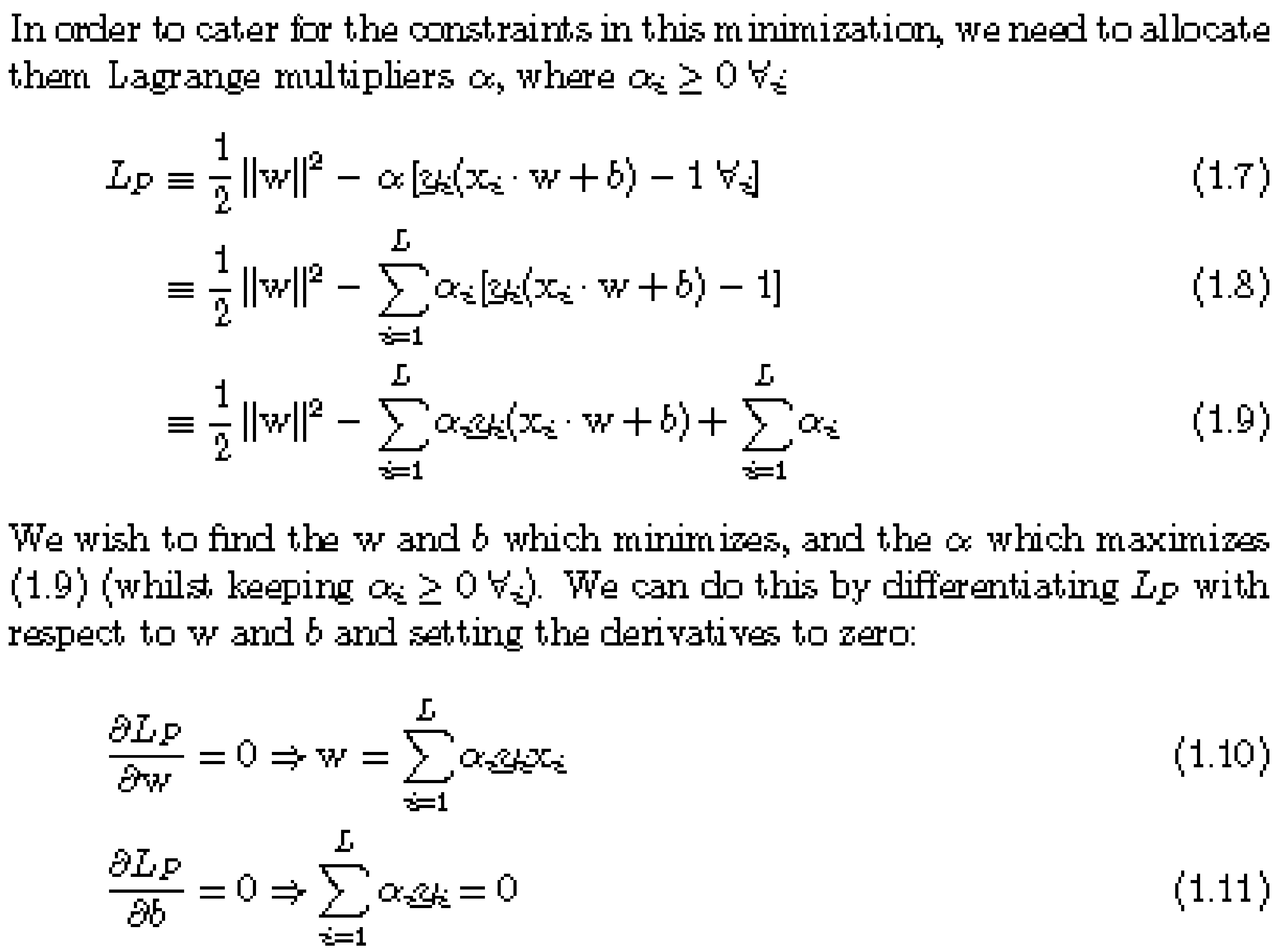
SVM classification: Hard-margin formulation



Minimizing $||w||$ is equivalent to minimizing $\frac{1}{2}||w||^2$ and the use of this term makes it possible to perform Quadratic Programming (QP) optimization later on. We therefore need to find:

$$\min \frac{1}{2}||w||^2 \quad \text{s.t.} \quad y_i(x_i \cdot w + b) - 1 \geq 0 \ \ \forall_i \tag{1.6}$$

SVM classification: Hard-margin formulation



In order to cater for the constraints in this minimization, we need to allocate them Lagrange multipliers α, where $\alpha_i \geq 0\ \forall_i$

$$L_P \equiv \frac{1}{2}\|\mathbf{w}\|^2 - \alpha\left[y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1\ \forall_i\right] \tag{1.7}$$

$$\equiv \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{L} \alpha_i\left[y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1\right] \tag{1.8}$$

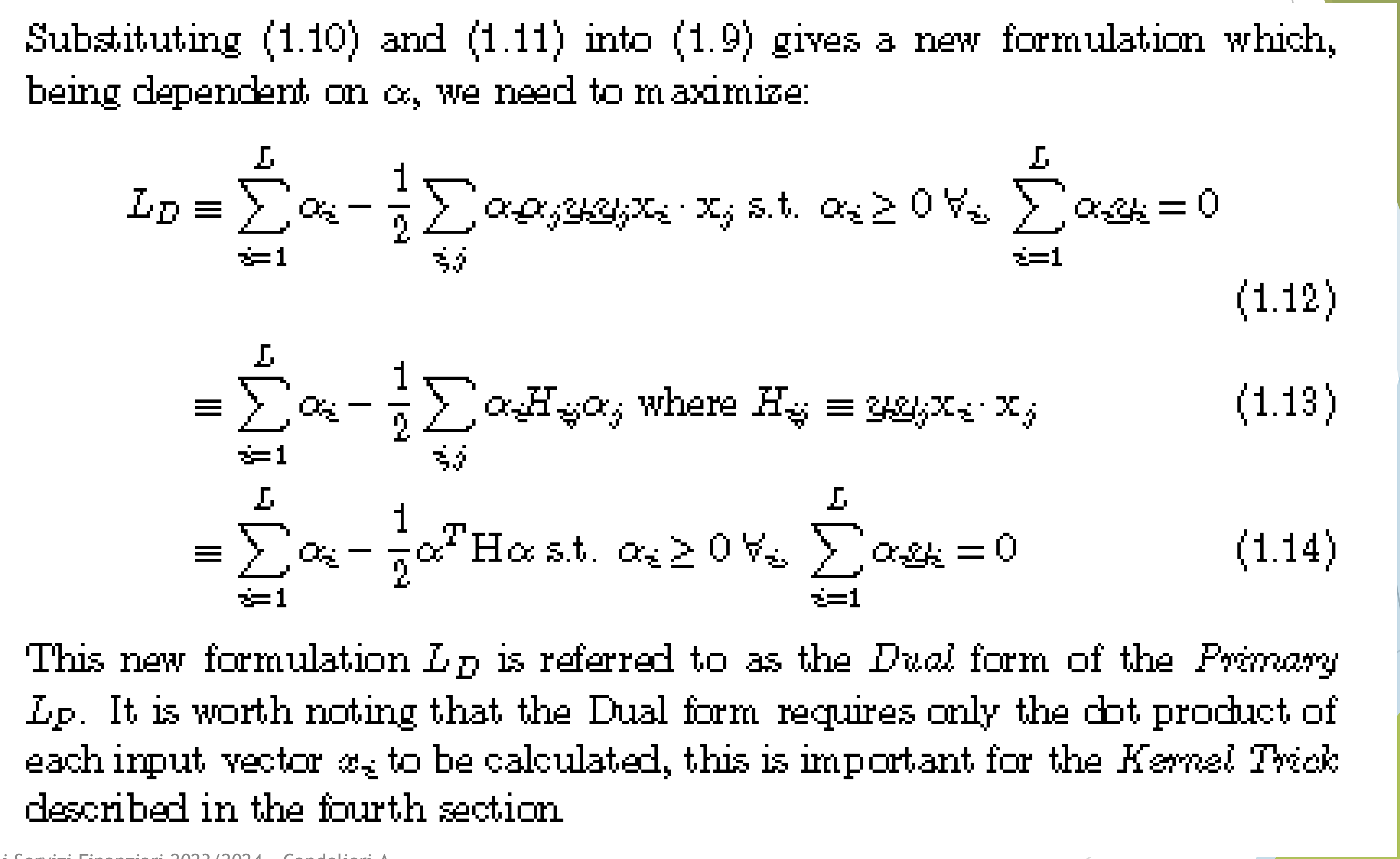
$$\equiv \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{L} \alpha_i y_i(\mathbf{x}_i \cdot \mathbf{w} + b) + \sum_{i=1}^{L} \alpha_i \tag{1.9}$$

We wish to find the $\mathbf{w}$ and b which minimizes, and the α which maximizes (1.9) (whilst keeping $\alpha_i \geq 0\ \forall_i$). We can do this by differentiating L_P with respect to $\mathbf{w}$ and b and setting the derivatives to zero:

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{L} \alpha_i y_i \mathbf{x}_i \tag{1.10}$$

$$\frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{i=1}^{L} \alpha_i y_i = 0 \tag{1.11}$$

SVM classification: Hard-margin formulation



Substituting (1.10) and (1.11) into (1.9) gives a new formulation which, being dependent on α, we need to maximize:

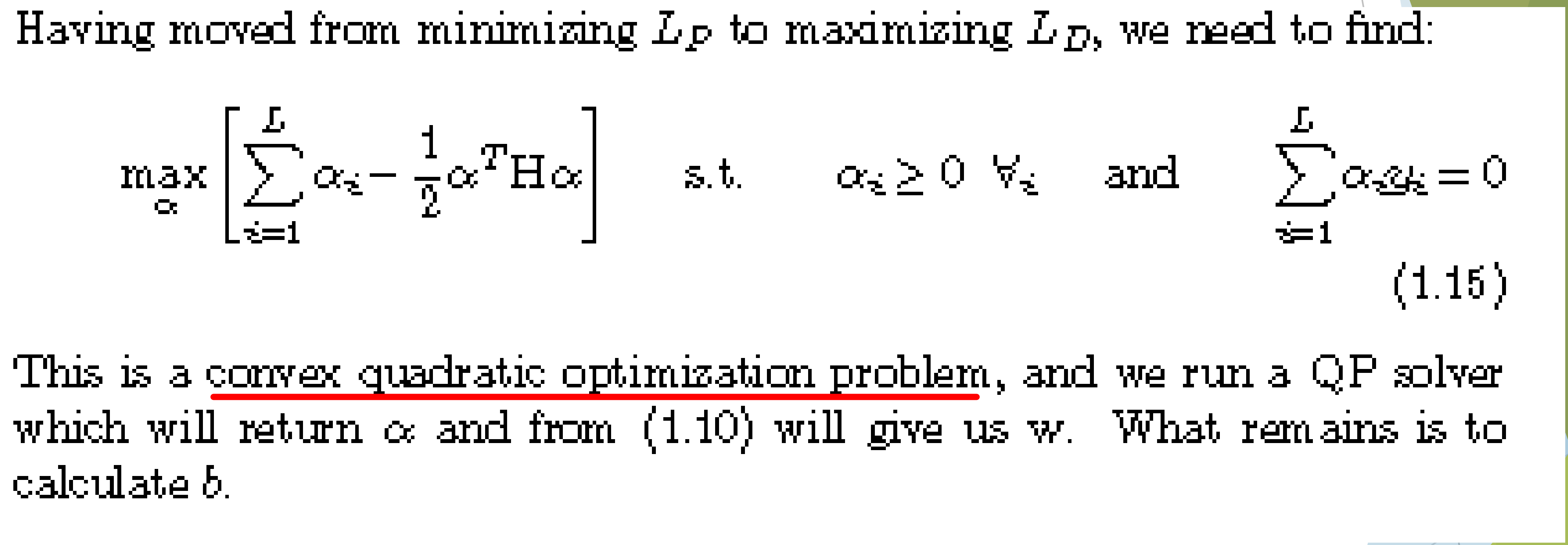
$$L_D \equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \text{ s.t. } \alpha_i \geq 0 \, \forall_i \, \sum_{i=1}^{L} \alpha_i y_i = 0 \tag{1.12}$$

$$\equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i H_{ij} \alpha_j \text{ where } H_{ij} \equiv y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{1.13}$$

$$\equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \alpha^T \mathbf{H} \alpha \text{ s.t. } \alpha_i \geq 0 \, \forall_i \, \sum_{i=1}^{L} \alpha_i y_i = 0 \tag{1.14}$$

This new formulation L_D is referred to as the *Dual* form of the *Primary* L_P. It is worth noting that the Dual form requires only the dot product of each input vector x_i to be calculated, this is important for the *Kernel Trick* described in the fourth section.

SVM classification: Hard-margin formulation



Having moved from minimizing L_P to maximizing L_D, we need to find:

$$\max_{\alpha}\left[\sum_{i=1}^{L}\alpha_i - \frac{1}{2}\alpha^T H \alpha\right] \quad \text{s.t.} \quad \alpha_i \geq 0 \;\; \forall_i \quad \text{and} \quad \sum_{i=1}^{L}\alpha_i y_i = 0 \tag{1.15}$$

This is a convex quadratic optimization problem, and we run a QP solver which will return α and from (1.10) will give us w. What remains is to calculate b.

SVM classification: Hard-margin formulation

Any data point satisfying (1.11) which is a Support Vector x_s will have the form:

$$y_s(\mathbf{x}_s \cdot \mathbf{w} + b) = 1$$

Substituting in (1.10):

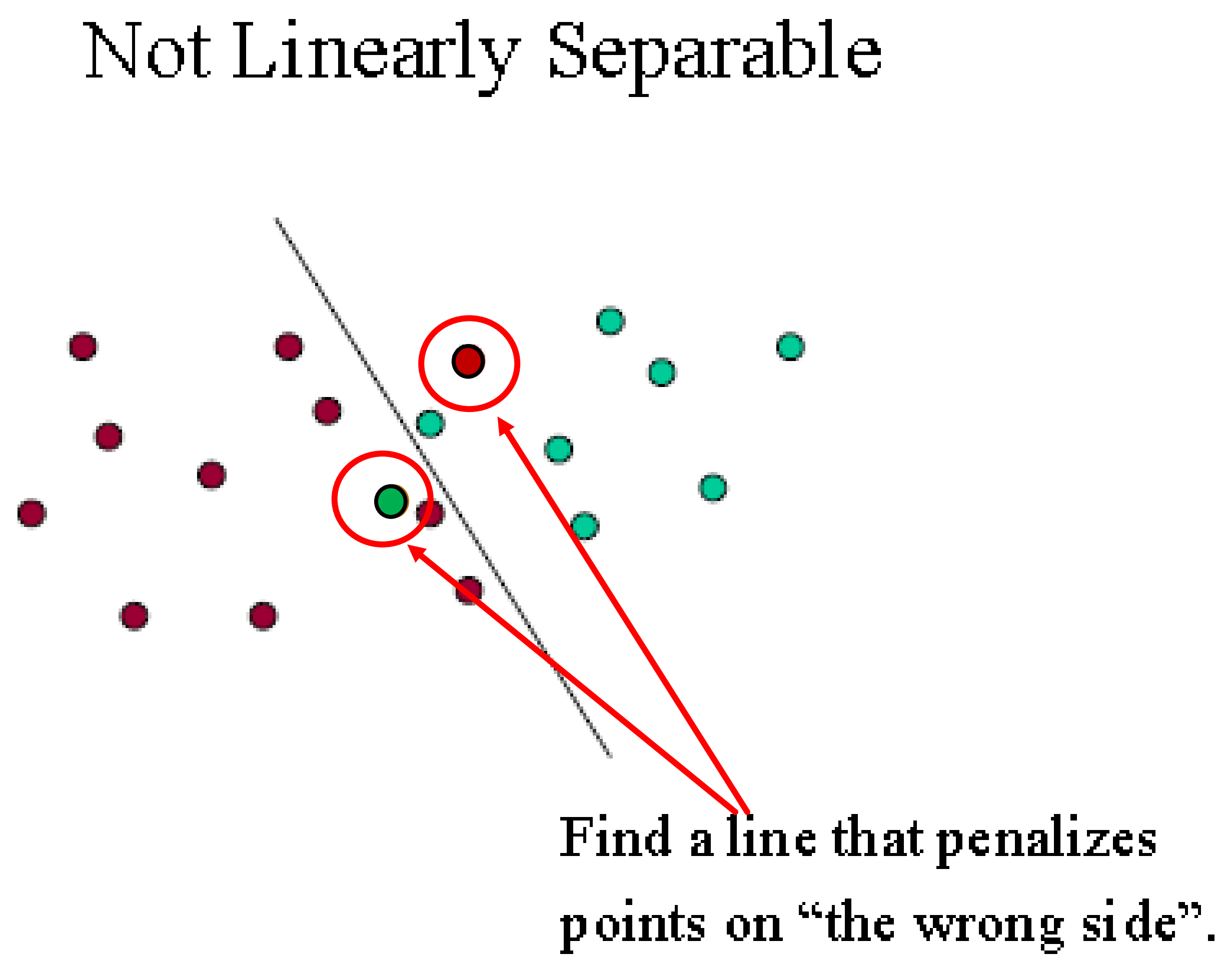
$$y_s\Big(\sum_{m \in S} \alpha_m y_m \mathbf{x}_m \cdot \mathbf{x}_s + b\Big) = 1$$

Where S denotes the set of indices of the Support Vectors. S is determined by finding the indices i where $\alpha_i > 0$. Multiplying through by y_s and then using $y_s^2 = 1$ from (1.1) and (1.2):

$$y_s^2\Big(\sum_{m \in S} \alpha_m y_m \mathbf{x}_m \cdot \mathbf{x}_s + b\Big) = y_s$$

$$b = y_s - \sum_{m \in S} \alpha_m y_m \mathbf{x}_m \cdot \mathbf{x}_s$$

Not Linearly Separable data



SVM classification: Soft-margin

In order to extend the SVM methodology to handle data that is not fully linearly separable, we relax the constraints for (1.1) and (1.2) slightly to allow for misclassified points. This is done by introducing a positive slack variable ξ_i, $i = 1, \ldots L$:

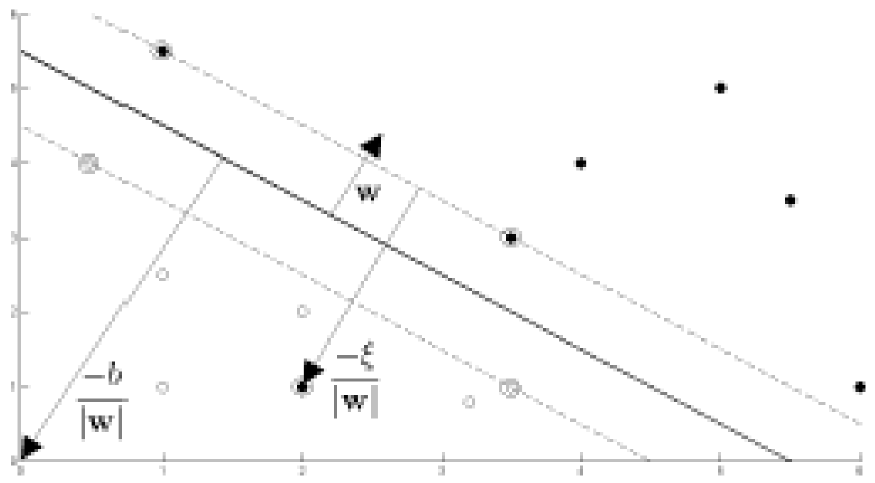
$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 \boxed{- \xi_i} \quad \text{for } y_i = +1 \tag{2.1}$$

$$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1 \boxed{+ \xi_i} \quad \text{for } y_i = -1 \tag{2.2}$$

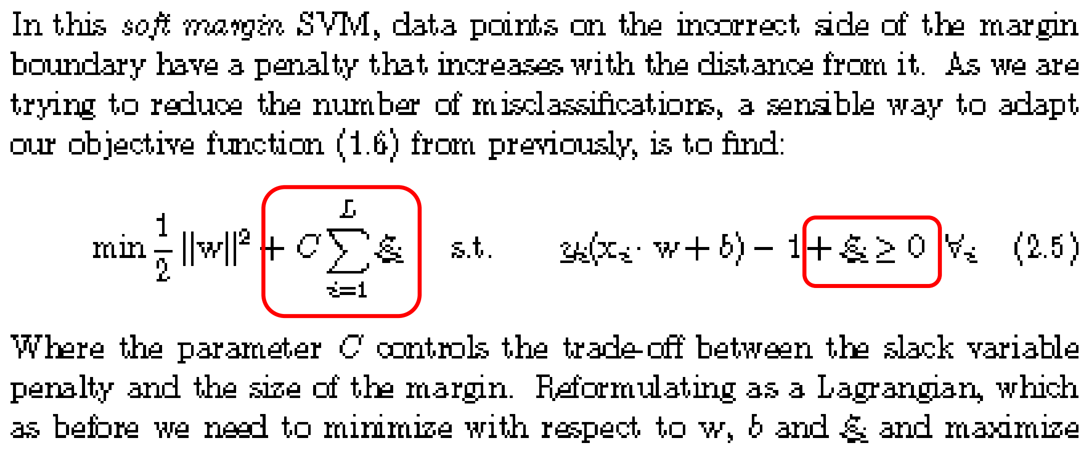
$$\boxed{\xi_i \geq 0 \ \forall_i} \tag{2.3}$$

Which can be combined into:

$$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \boxed{+ \xi_i \geq 0} \quad \text{where} \quad \xi_i \geq 0 \ \forall_i \tag{2.4}$$



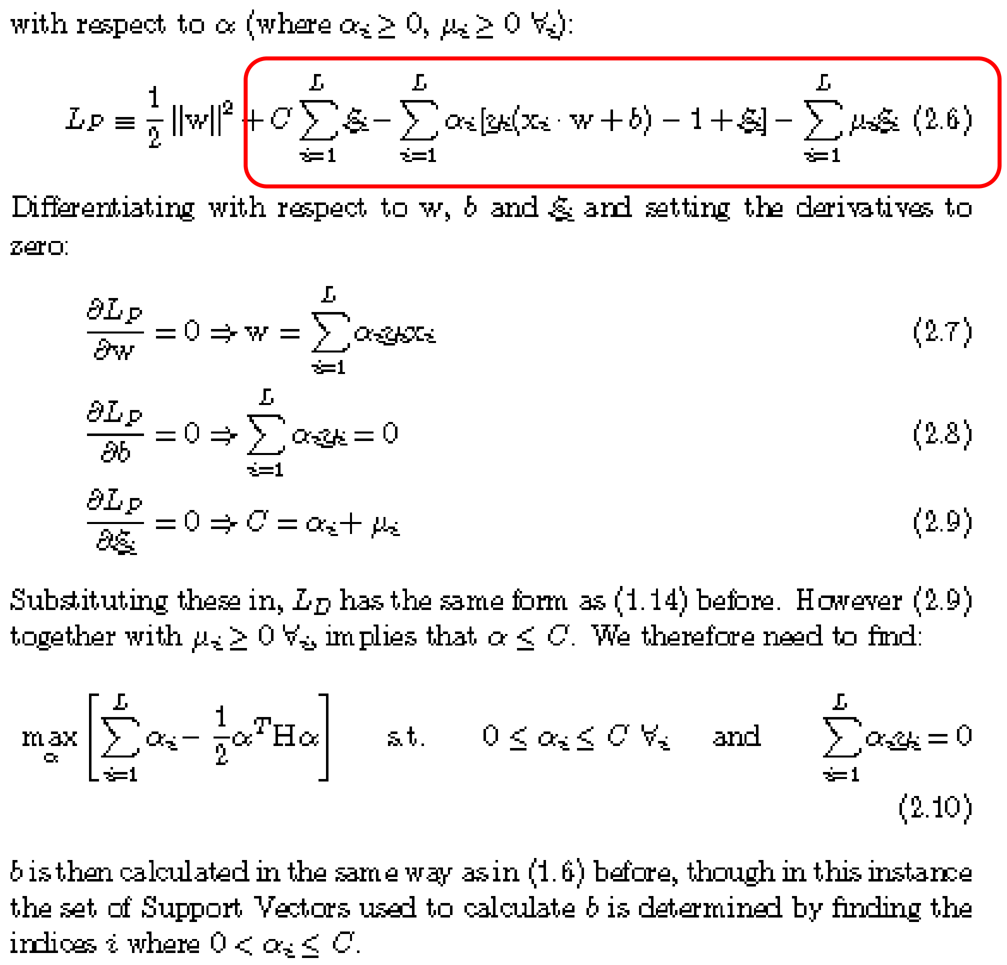
SVM classification: Soft-margin



In this *soft margin* SVM, data points on the incorrect side of the margin boundary have a penalty that increases with the distance from it. As we are trying to reduce the number of misclassifications, a sensible way to adapt our objective function (1.6) from previously, is to find:

$$\min \frac{1}{2}\|\mathbf{w}\|^2 \boxed{+ C\sum_{i=1}^{L}\xi_i} \quad \text{s.t.} \quad y_i(\mathbf{x}_i\cdot \mathbf{w} + b) - 1 \boxed{+\xi_i \geq 0} \ \forall_i \qquad (2.5)$$

Where the parameter C controls the trade-off between the slack variable penalty and the size of the margin. Reformulating as a Lagrangian, which as before we need to minimize with respect to $\mathbf{w}$, b and ξ_i and maximize



with respect to α (where $\alpha_i \geq 0$, $\mu_i \geq 0 \ \forall_i$):

$$L_P \equiv \frac{1}{2}\|\mathbf{w}\|^2 \boxed{+ C\sum_{i=1}^{L}\xi_i - \sum_{i=1}^{L}\alpha_i\left[y_i(\mathbf{x}_i\cdot\mathbf{w}+b) - 1 + \xi_i\right] - \sum_{i=1}^{L}\mu_i\xi_i \quad (2.6)}$$

Differentiating with respect to $\mathbf{w}$, b and ξ_i and setting the derivatives to zero:

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{L}\alpha_i y_i \mathbf{x}_i \qquad (2.7)$$

$$\frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{i=1}^{L}\alpha_i y_i = 0 \qquad (2.8)$$

$$\frac{\partial L_P}{\partial \xi_i} = 0 \Rightarrow C = \alpha_i + \mu_i \qquad (2.9)$$

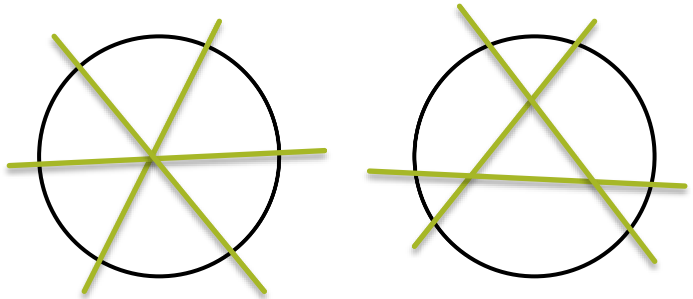
Substituting these in, L_D has the same form as (1.14) before. However (2.9) together with $\mu_i \geq 0 \ \forall_i$ implies that $\alpha \leq C$. We therefore need to find:

$$\max_{\alpha}\left[\sum_{i=1}^{L}\alpha_i - \frac{1}{2}\alpha^T \mathbf{H}\alpha\right] \quad \text{s.t.} \quad 0 \leq \alpha_i \leq C \ \forall_i \quad \text{and} \quad \sum_{i=1}^{L}\alpha_i y_i = 0 \qquad (2.10)$$

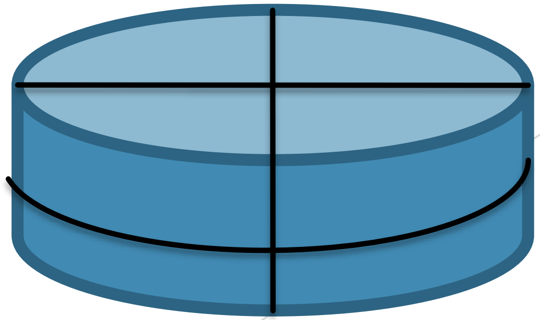
b is then calculated in the same way as in (1.6) before, though in this instance the set of Support Vectors used to calculate b is determined by finding the indices i where $0 < \alpha_i \leq C$.

Dealing with non-linearity

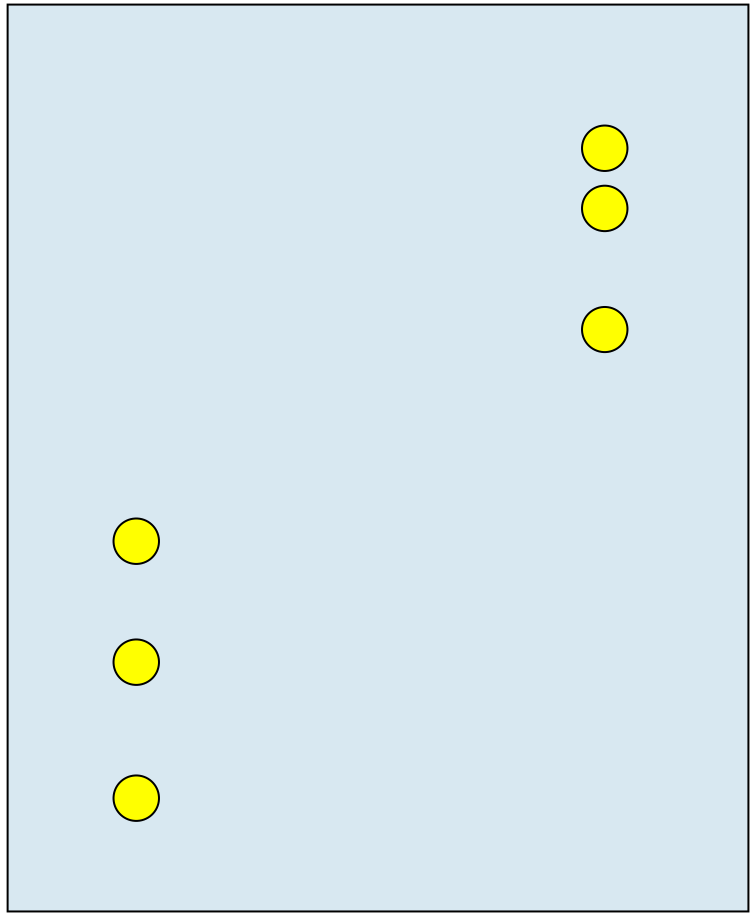
Can you divide a cake in 8 equal slices using only three linear cuts?



It is impossible in 2D, but in 3D …

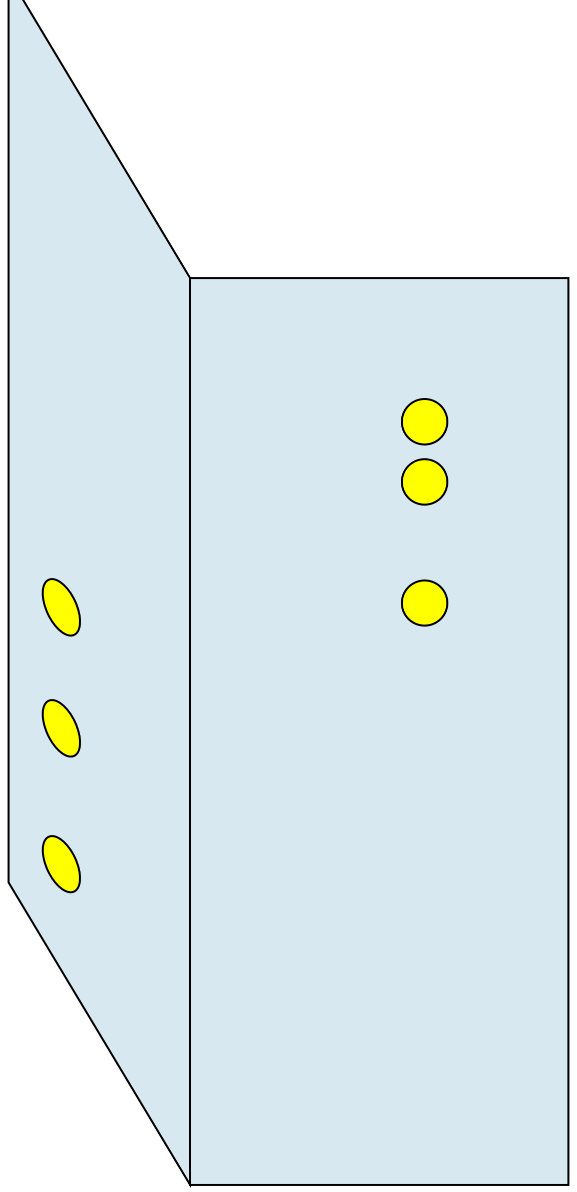


and…

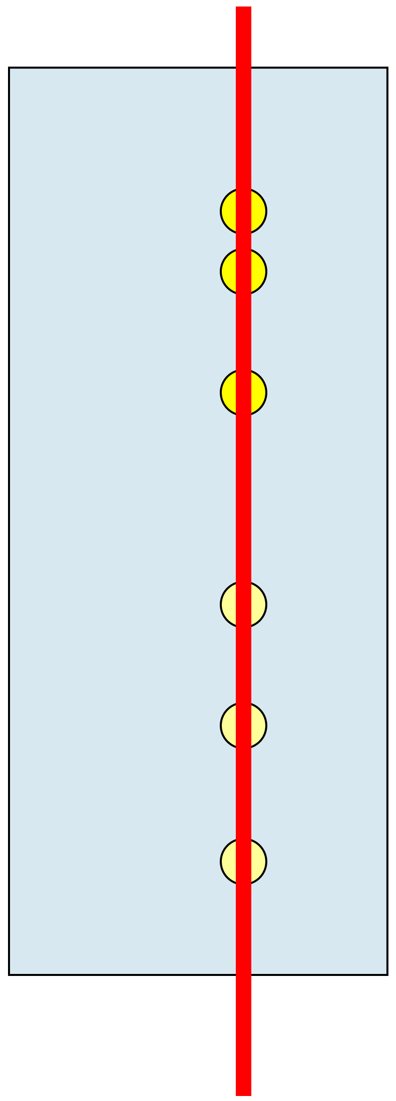


Does a line passing through the 6 points exist?

and…



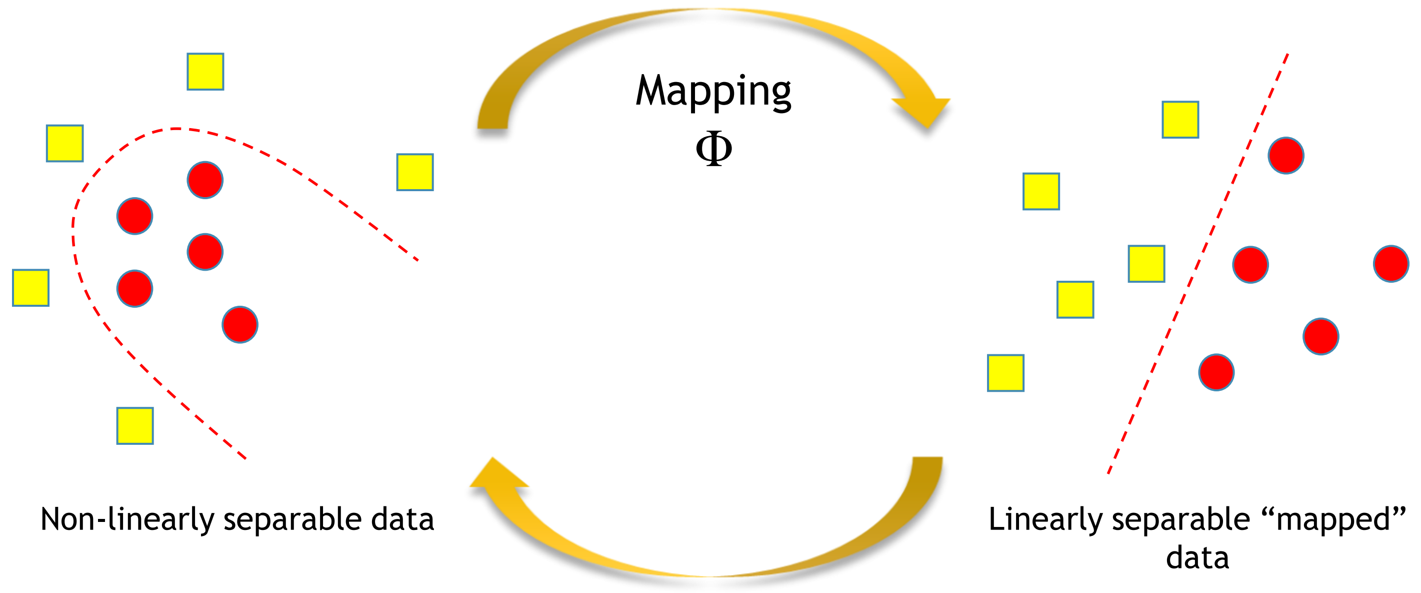
and…



…therefore…

"Marty, you're just not thinking fourth dimensionally"

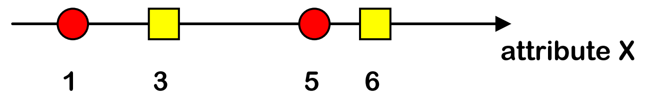
Trasformazioni spaziali



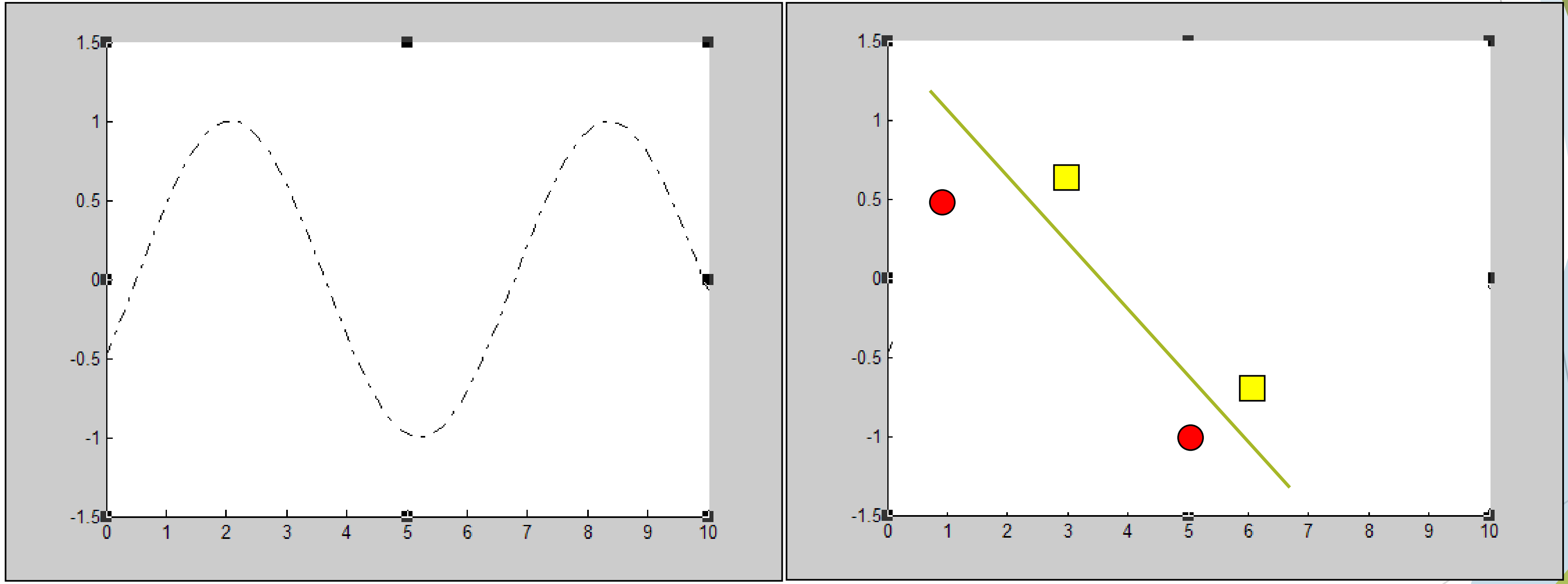
Move from a space in which instances are not linearly separable (aka *Input Space*) to a space in which (hopefully) they are (aka *Feature Space*), by using a Mapping function

The linear separation hyper-plane in the Feature Space is equivalent to a non-linear separation hyper-plane in the original Attribute Space

An example

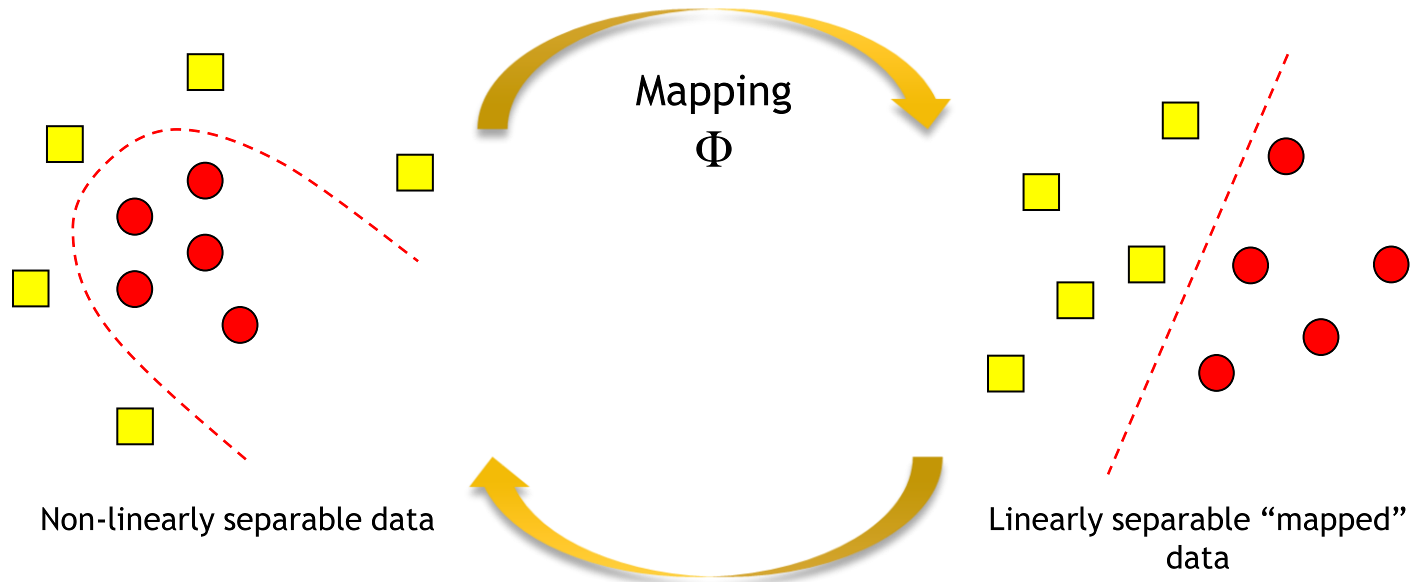


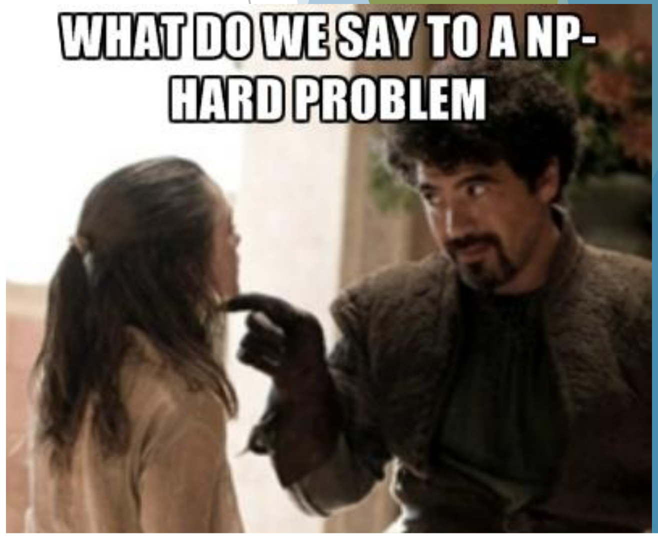
Non-linearly separable data



Non-linear transformation: Y=sin(X-0.5)

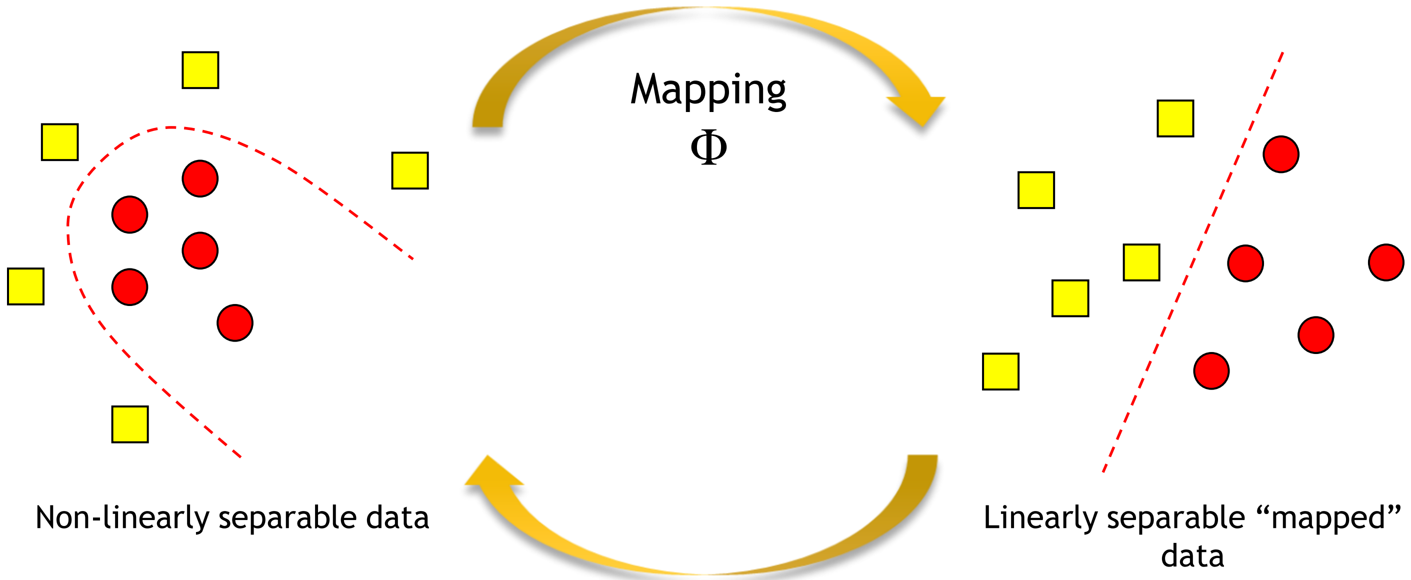
Spatial transformation

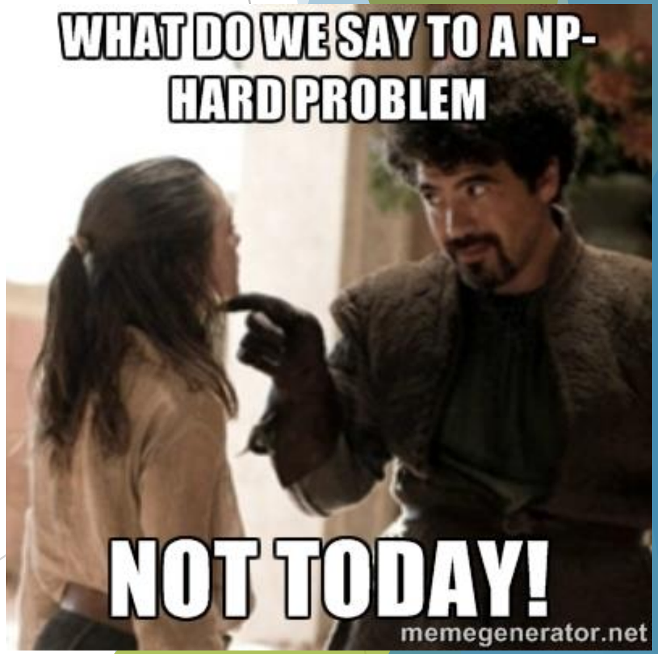




Finding the optimal mapping is NP-Hard…

Spatial transformation





Finding the optimal mapping is NP-Hard…

Typical Mapping

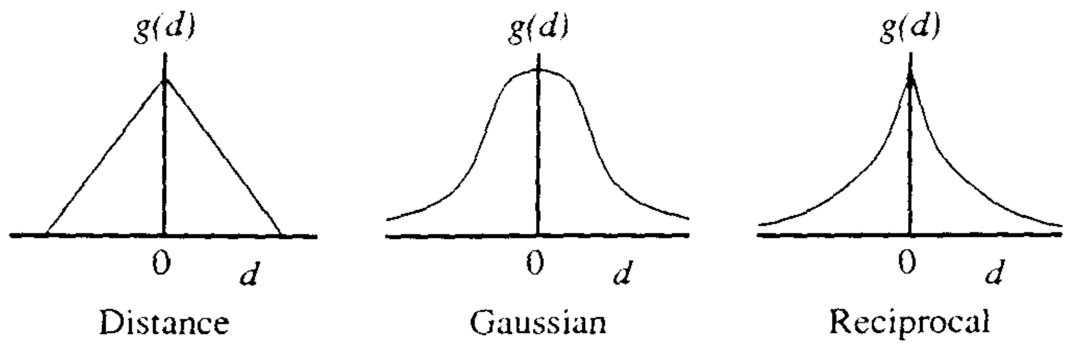
Polynomial

$$K(x,y) = (<x,y> + c)^n$$

Radial Basis Function (RBF)

$$K(x,y) = e^{-\gamma \|x-y\|^2}$$

Radial Basis Functions

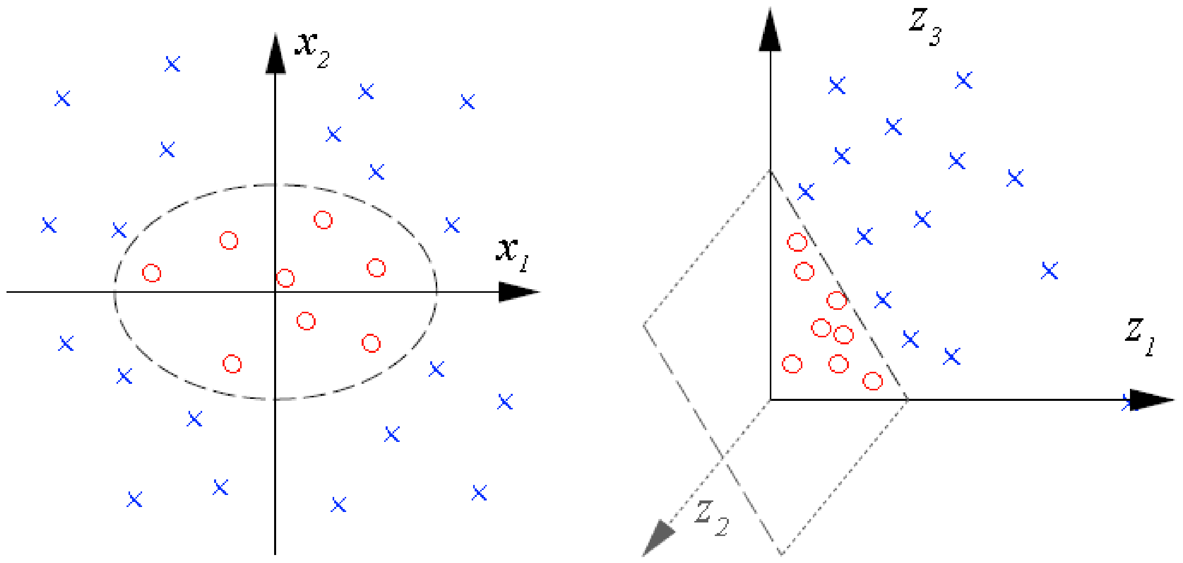


d is the distance between a learned pattern and a new pattern.

An example of (polynomial mappng)

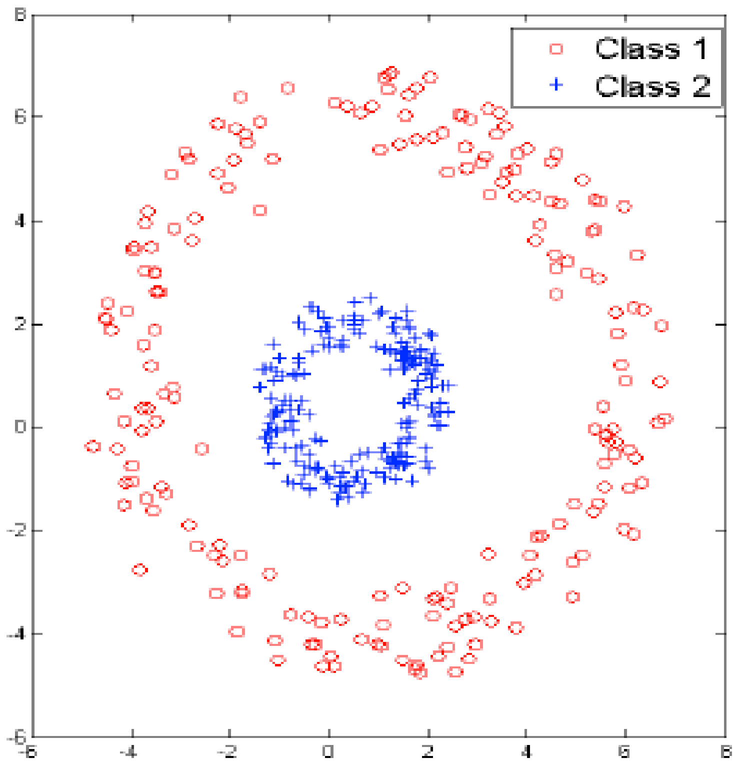
SVMs : polynomial mapping

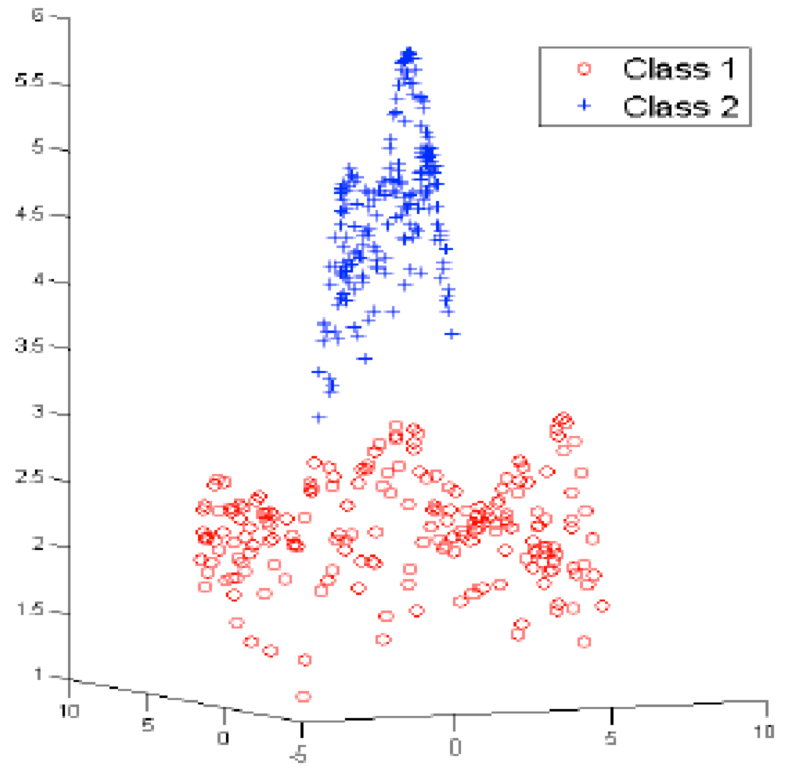
$$\Phi : R^2 \rightarrow R^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt(2)x_1x_2, x_2^2)$$



The RBF (Radial Basis Function) kernel

$$k(\mathbf{x}_i, \mathbf{x}_j) = e^{-\left(\frac{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{2\sigma^2}\right)}$$





SVM with kernel: non-linear classification

We can rewrite all the SVM equations we saw before, but with the $w = \sum_{i=1}^{m} \alpha_i \Phi(x_i)$ equation:

- **Decision function:**

$$f(x) = \sum_i \alpha_i \Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}) + b$$

$$= \sum_i \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b$$

- **Dual formulation:**

$$\min P(\boldsymbol{w}, b) = \underbrace{\frac{1}{2} \| \sum_{i=1}^{m} \alpha_i \Phi(x_i) \|^2}_{\text{maximize margin}} + \underbrace{C \sum_i H_1[y_i f(\boldsymbol{x}_i)]}_{\text{minimize training error}}$$

B

SVM with kernel: non-linear classification

- **Dual formulation:**

$$\min_{\alpha} D(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i,j} \alpha_i\, \alpha_j\, \Phi(\boldsymbol{x}_i)\cdot\Phi(\boldsymbol{x}_j) - \sum_i y_i\, \alpha_i \quad \text{s.t.} \begin{cases} \sum_i \alpha_i = 0 \\ 0 \leq y_i\, \alpha_i \leq C \end{cases}$$

- **Dual Decision function:**

$$f(x) = \sum_i \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b$$

- **Kernel function** $K(\cdot,\cdot)$ is used to make (implicit) nonlinear feature map, e.g.
  - Polynomial kernel: $K(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}\cdot\boldsymbol{x}' + 1)^d.$
  - RBF kernel: $K(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\gamma\|\boldsymbol{x} - \boldsymbol{x}'\|^2).$

SVM for regression

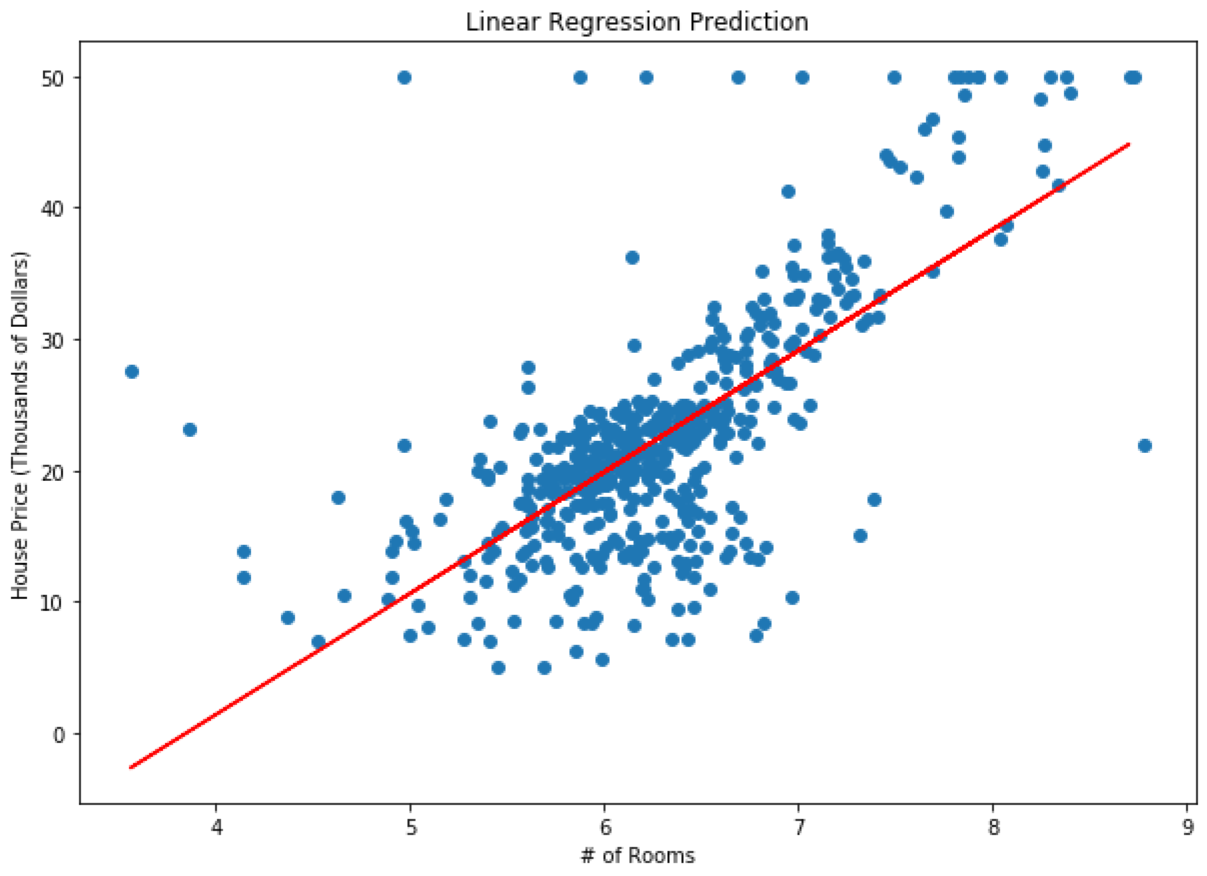
- To better understand SVM regression learning, we have to start with the simplest regression algorithm: **linear regression**

- The goal is to find the best linear approximation for a set of available data…

- …that is: searching for the coefficients of the linear regression minimizing a measure of error (usually the sum of squared errors):

$$\min_{w \in \mathbb{R}^d} \sum_{i=1}^{N} \left(y^{(i)} - w \cdot x^{(i)} \right)^2$$

Let us consider the simplest case



Why is SVM regression different?

- In SVM regression the goal is to minimize the L2-norm of the coefficients vector w (instead of the sum of squared error) while the error term is added into a consraint

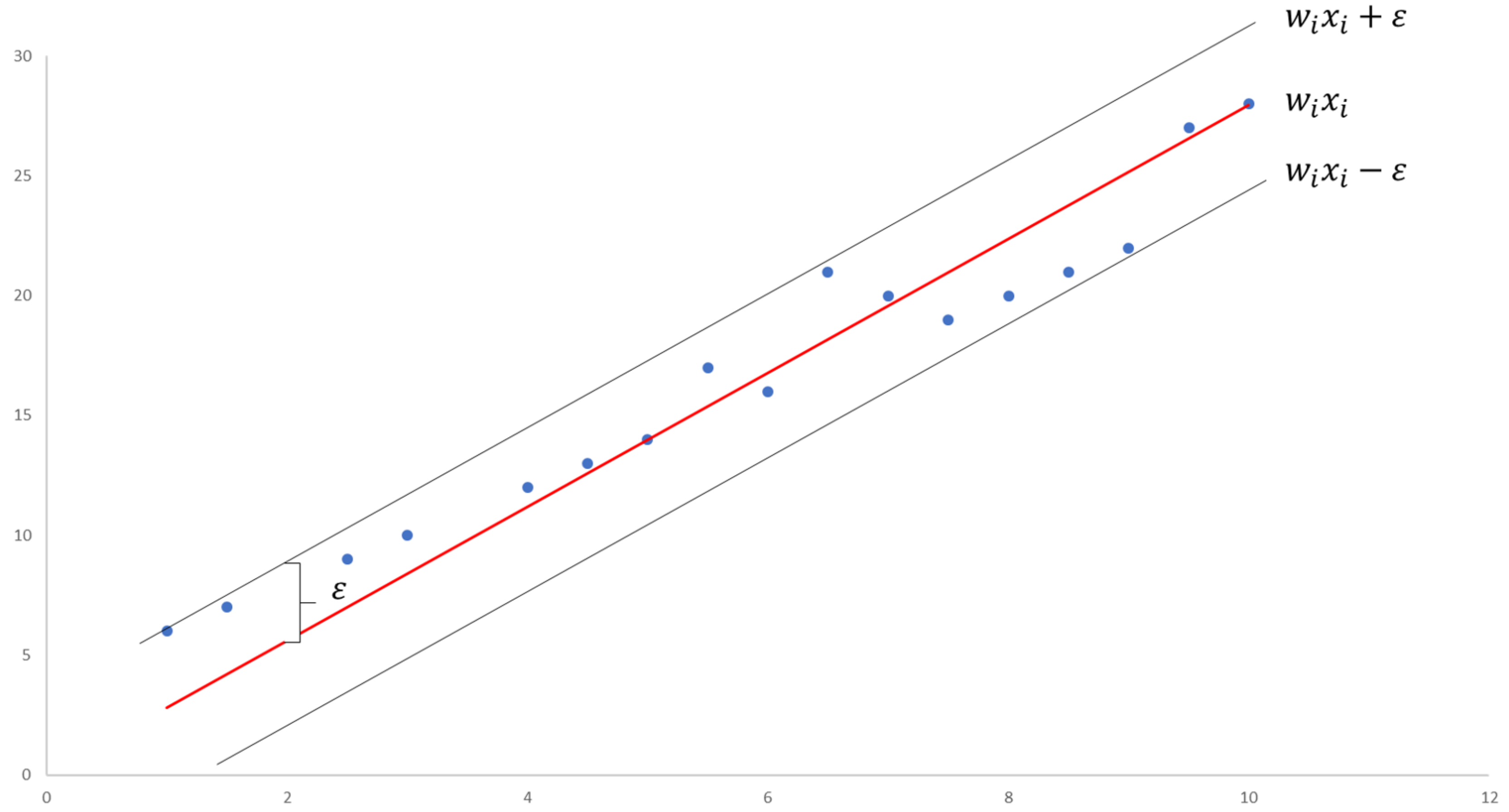
- Constraint: the absolute error must be less than or equal to a specified value $\varepsilon > 0$

$$\min_{w \in \mathbb{R}^d} \frac{1}{2} \|w\|^2$$

$$s.t. \left| y^{(i)} - w \cdot x^{(i)} \right| \leq \varepsilon \; \forall \; i = 1, \dots, N$$

- Consequently, errors less than ε are not considered → it is also said that SVM regression is **ε-insensitive**

SVM regression: a simple example

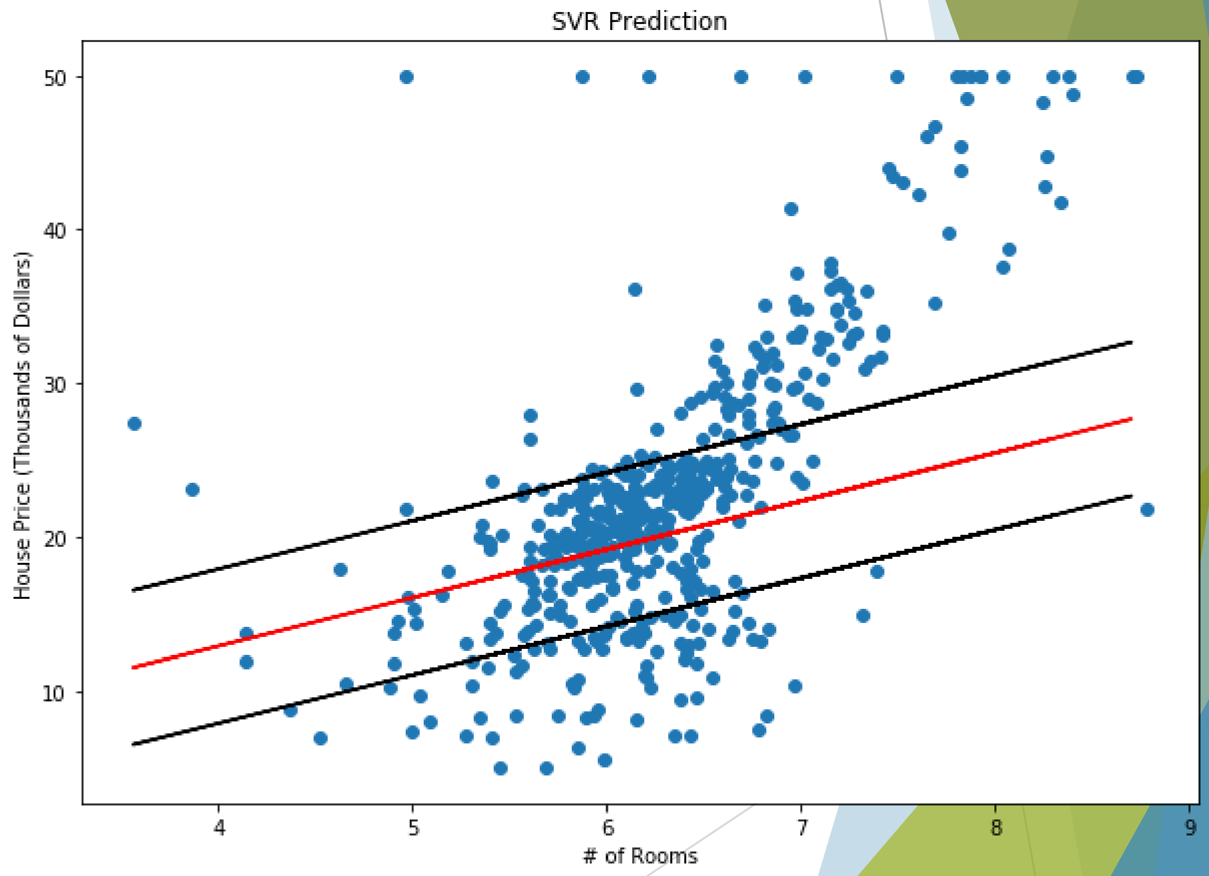


SVM regression: a more concrete simple example…

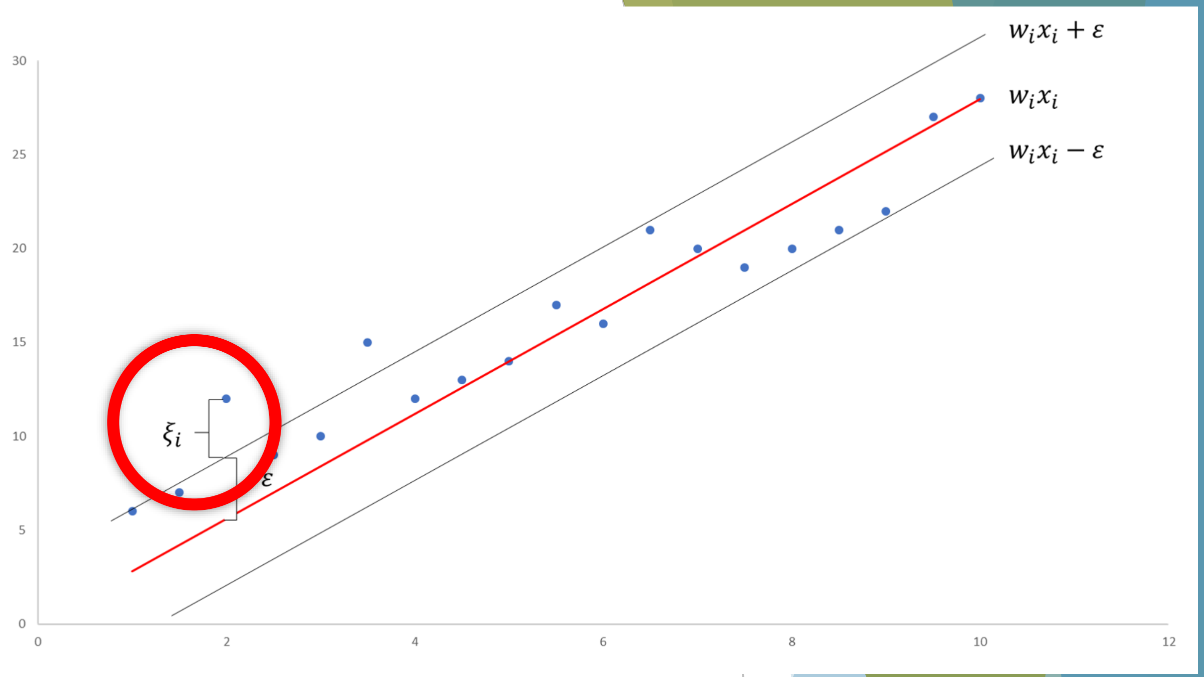
- The constrained objective function is minimized as best as possible but some of the data points can fall outside the margins

- We need to account for the possibility of errors that are larger than ε

- Thus, we introduce slack variables



SVM regression



- With tha slack variables the problem becomes

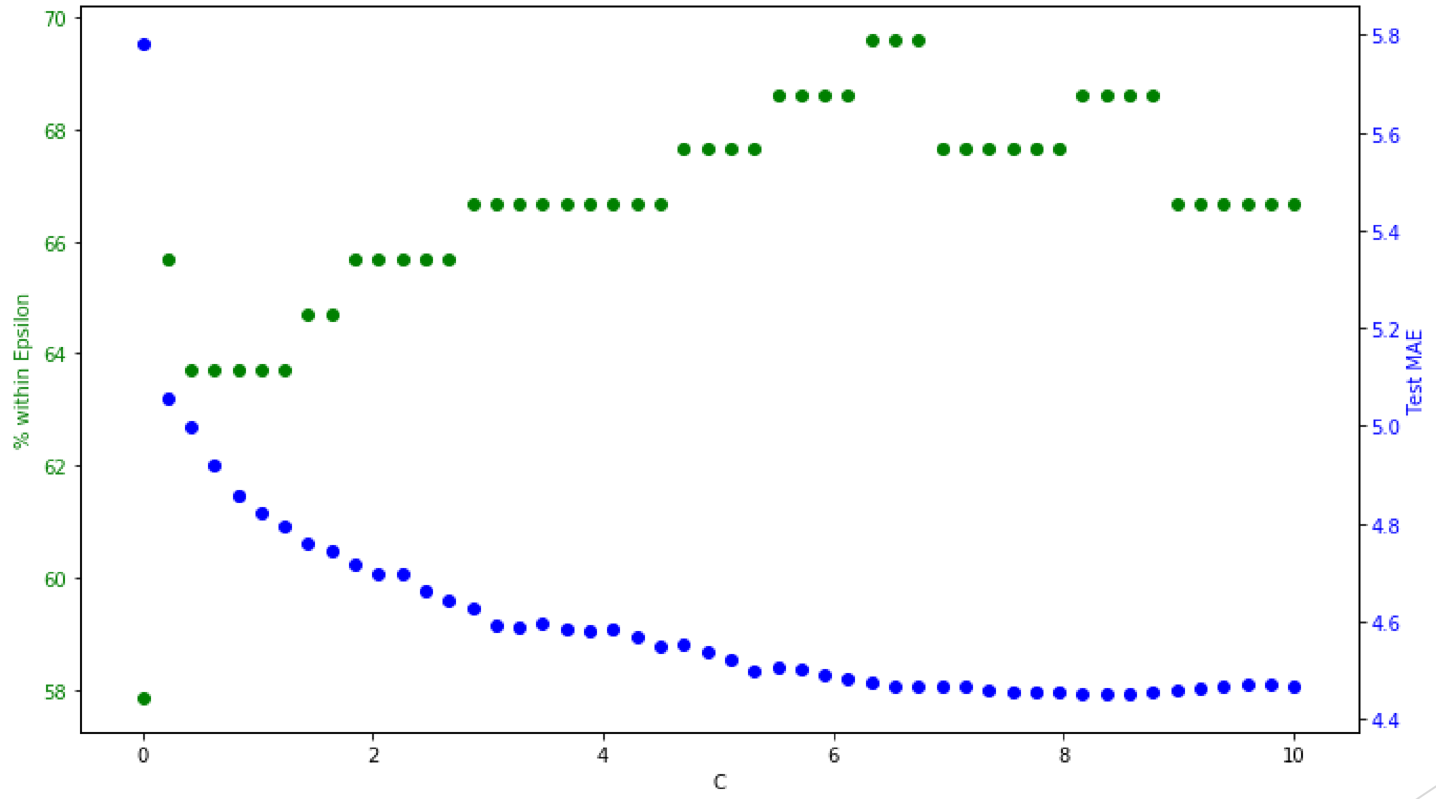
$$\min_{w \in \mathbb{R}^d} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\left|\xi^{(i)}\right|$$

$$s.t.\ \left|y^{(i)} - w \cdot x^{(i)}\right| \le \varepsilon + \left|\xi^{(i)}\right|\ \forall\, i = 1, \dots, N$$

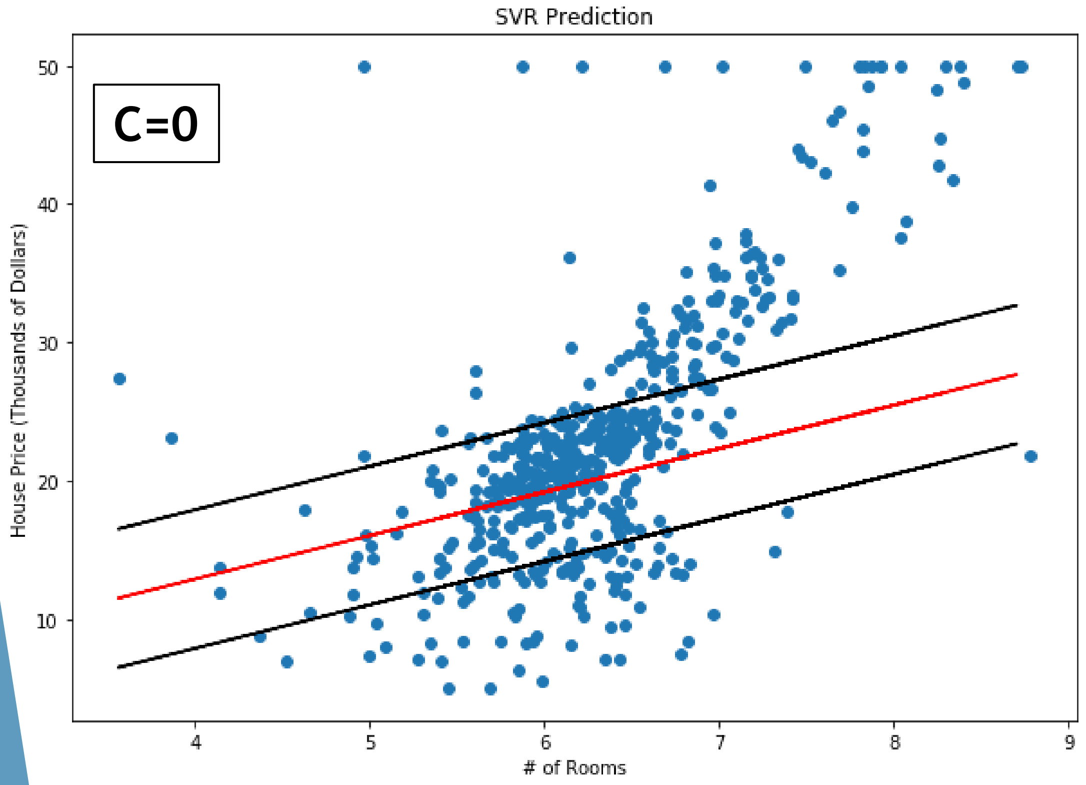
- implying the adoption of the hyperparameter ***C***, dealing with the trade off between the model complexity (flatness) and the degree to which deviations larger than are tolerated in optimization formulation for example
- as C increases, tolerance for points outside of ε increases. As C approaches 0, the tolerance approaches 0 and the equation collapses into the previous simplified one

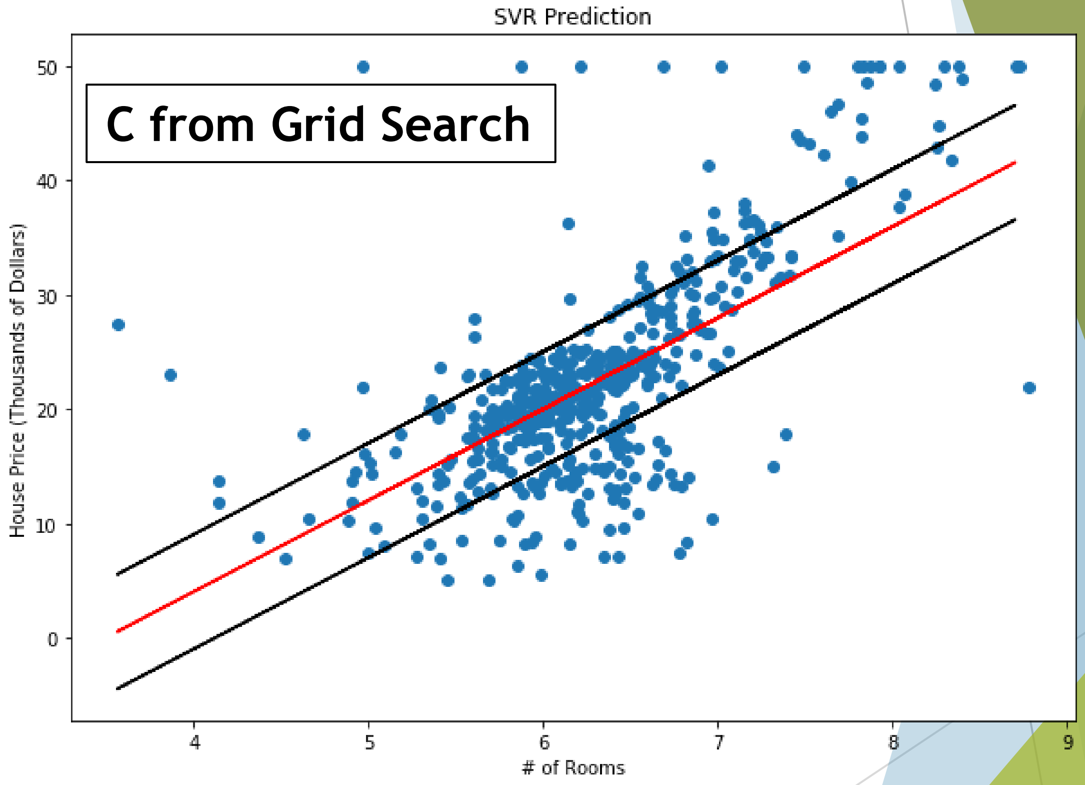
Finding the best value of *C*

- Grid Search (as well as Random Search) is a search strategy very simple to implement



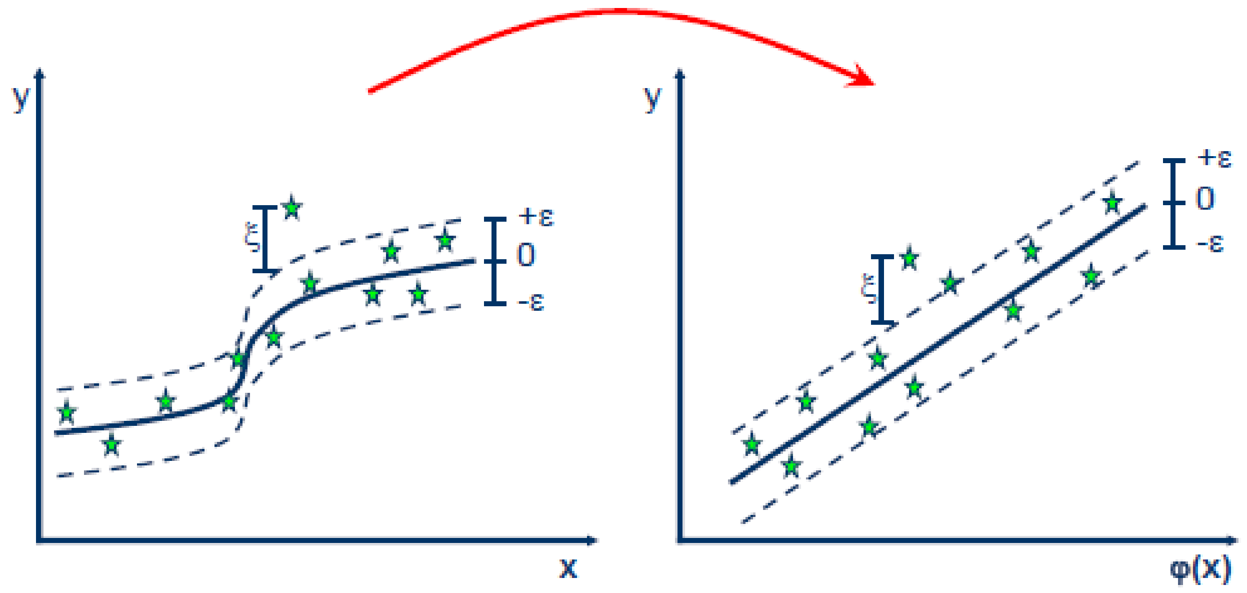
An example about the role of *C*





Dealing with non-linearities

- *Kernel trick*: a kernel function maps the data into a higher dimensional feature space to make it possible to perform the linear separation.



The kernel trick

- Instead of $x^{(i)}$ we have to consider consider $\varphi\left(x^{(i)}\right)$ and consider the previous optimization problem

- But searching for the best "mapping" $\varphi(x)$ is NP-hard

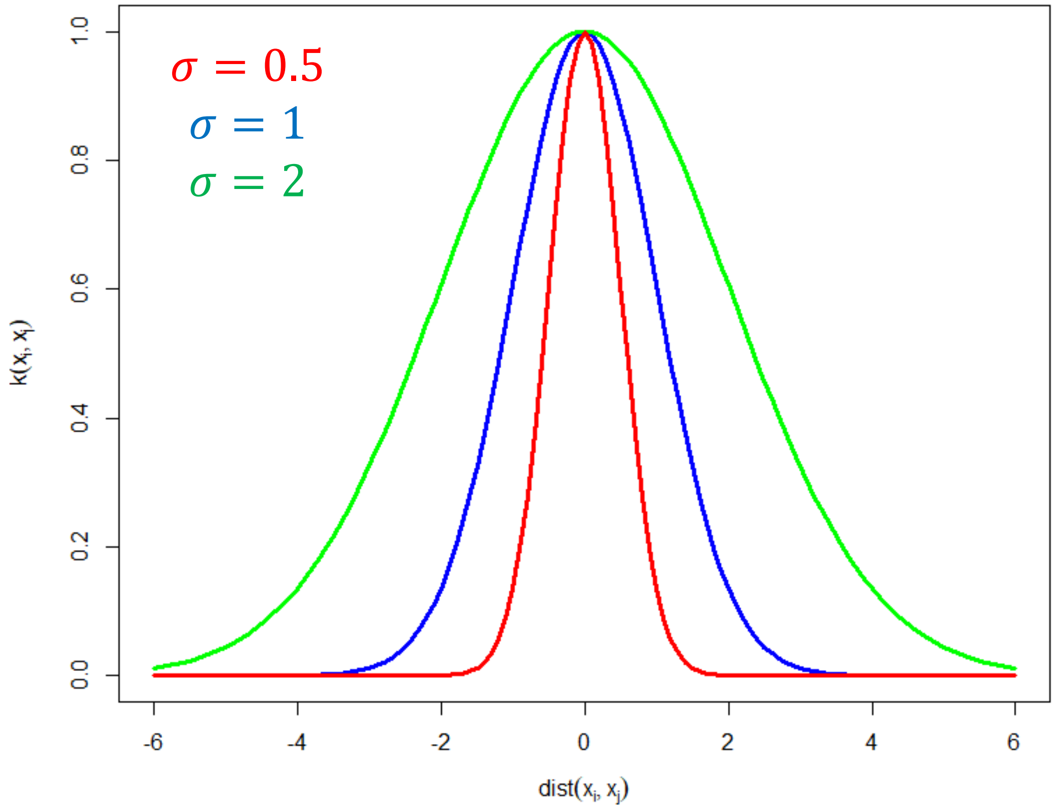
- Kernel-trick allows to use – implicitly – some "good families" of $\varphi(x)$

- One of the most widely used kernel is the Guassian / Radial Basis Function:

$$k\left(x^{(i)}, x^{(j)}\right) = e^{-\frac{\left\|x^{(i)}-x^{(j)}\right\|^2}{2\sigma^2}}$$

The role of kernel

- Kernel is a similarity measure in the feature speace induced by the implicit mapping $\varphi(x)$
- The kernel's hyperparameter σ allows for tuning this similarity



Finally,…

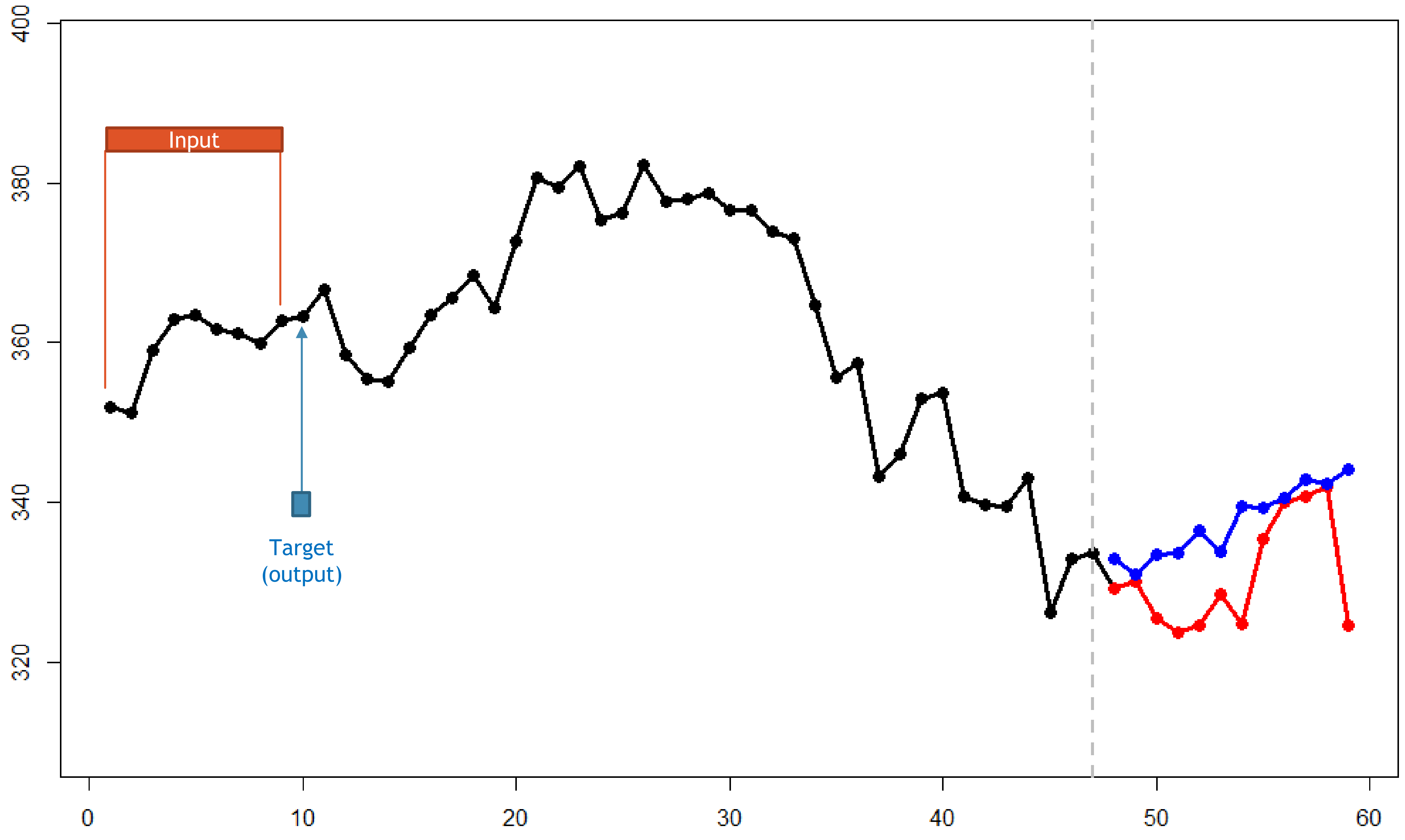
- *Just a little bit of "black magic"…*
- The (primal) optimization problem can be transformed into its dual version, with solution given by:

$$\hat{y}(x) = \sum_{i=1}^{\nu} \left(\alpha^{(i)} - \alpha^{*(i)}\right) k\left(x^{(i)}, x\right)$$

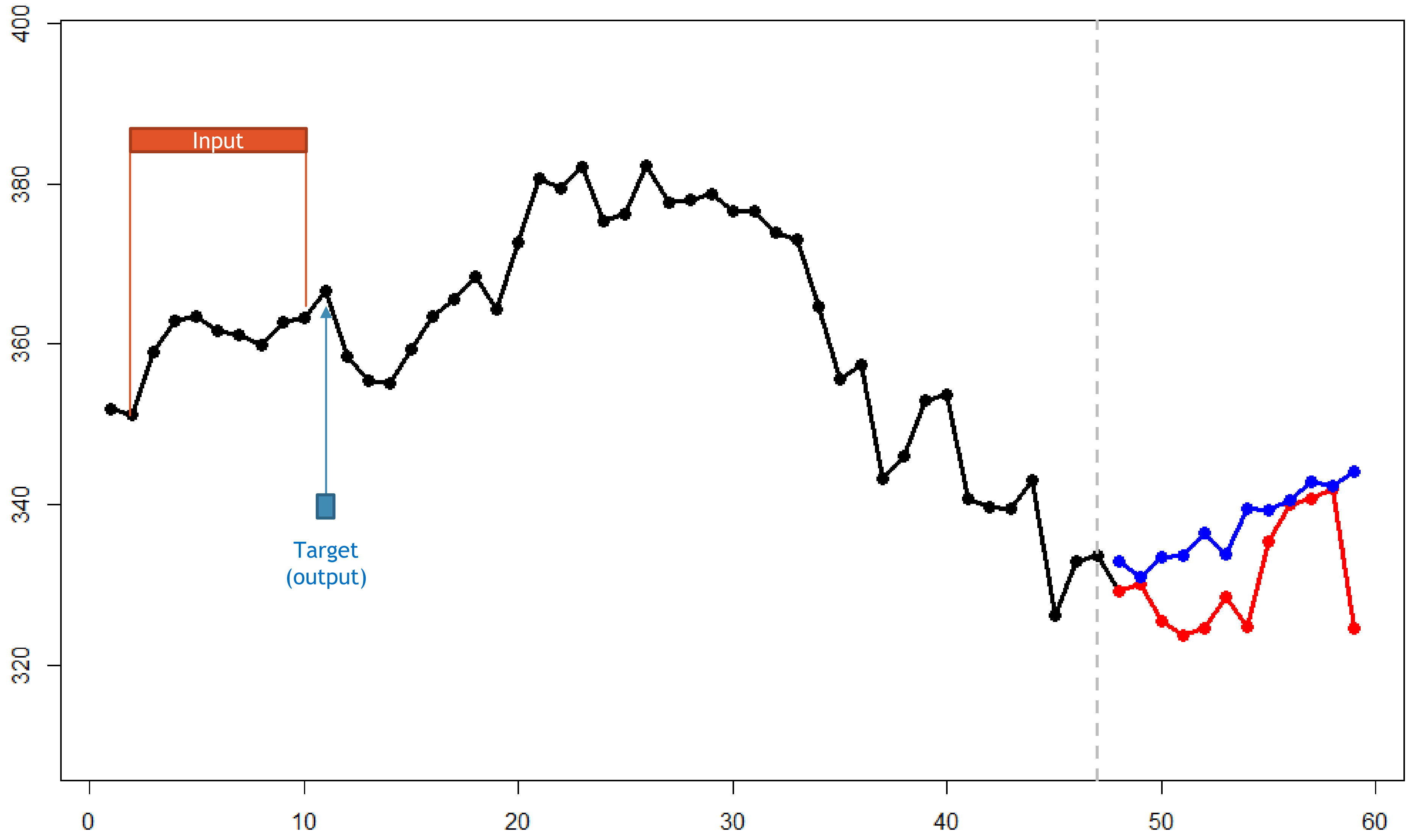
- where ν is the number of support vectors and $\alpha^{(i)}$ and $\alpha^{*(i)}$ are the dual variables related to, respectively, the constraints:

$$\left.\begin{aligned} y^{(i)} - w \cdot x^{(i)} &\leq \varepsilon + \left|\xi^{(i)}\right| \\ w \cdot x^{(i)} - y^{(i)} &\geq \varepsilon + \left|\xi^{(i)}\right| \end{aligned}\right\} \quad \left|y^{(i)} - w \cdot x^{(i)}\right| \leq \varepsilon + \left|\xi^{(i)}\right|$$

From SVM regression to SVM forecasting

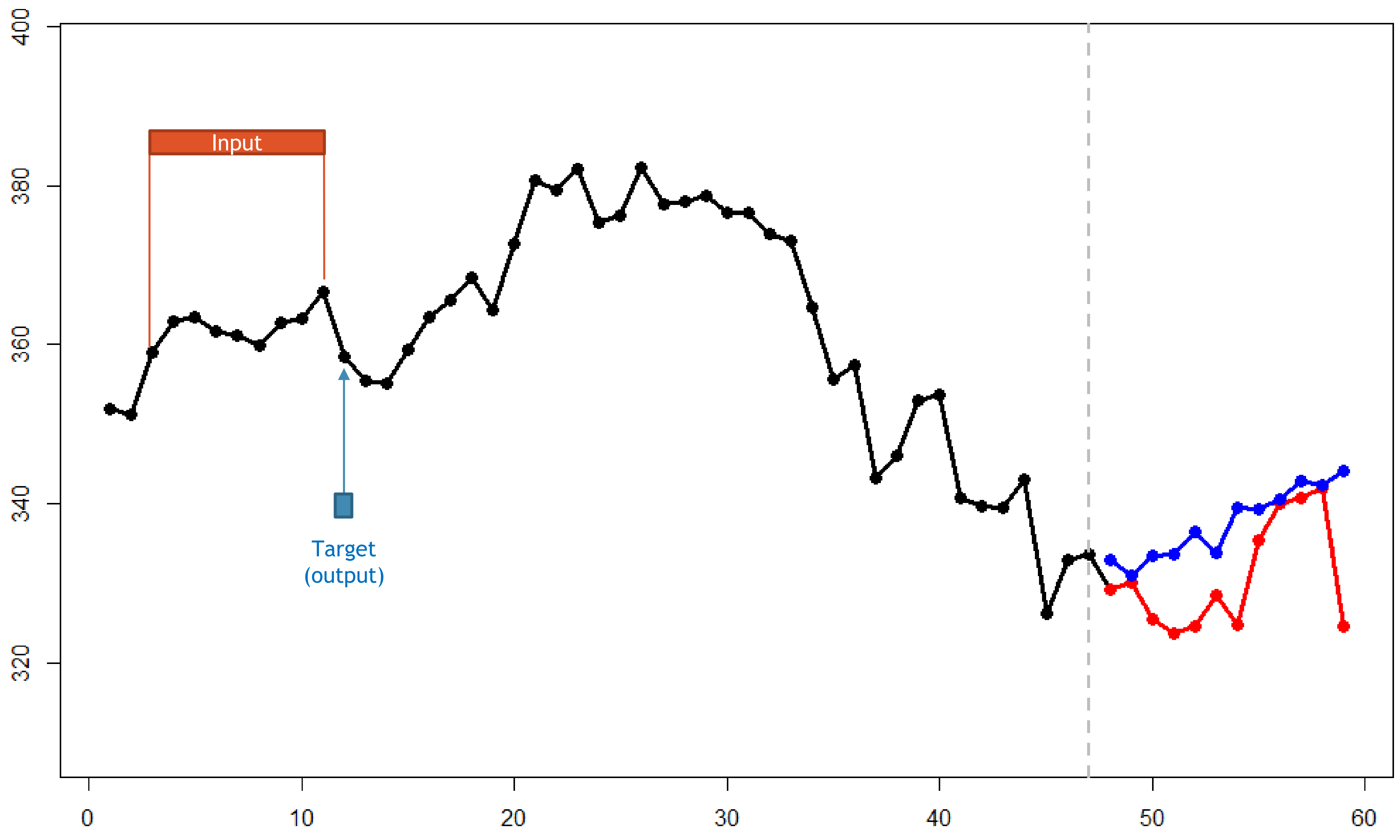


From SVM regression to SVM forecasting



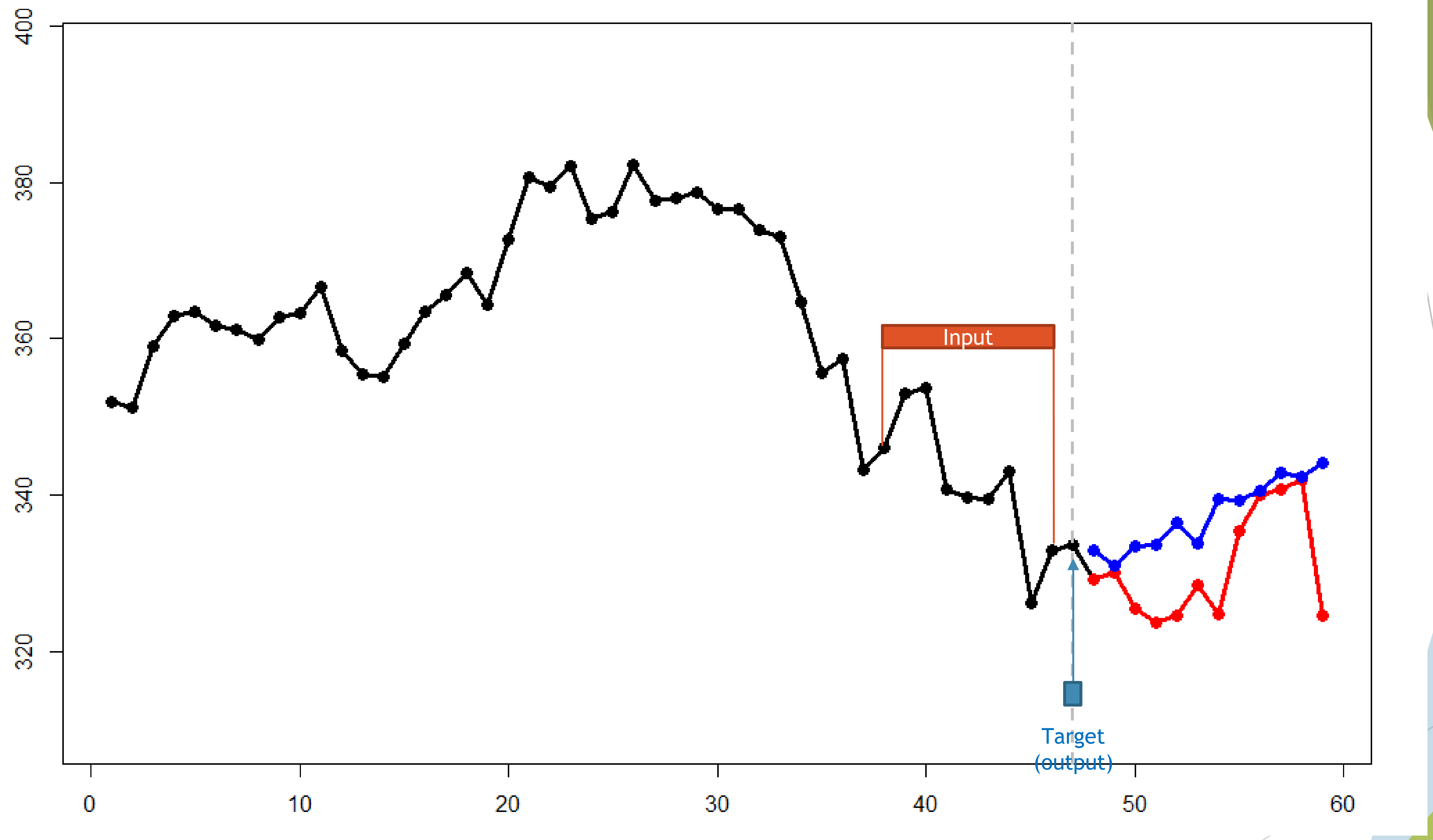
First data 🟦
Second data 🟦

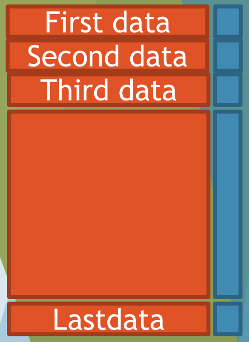
From SVM regression to SVM forecasting



First data 🟦
Second data 🟦
Third data 🟦

From SVM regression to SVM forecasting





Limitations of SVM forecasting

- SVM forecasting is, natively, one-step forecasting
- Only recursive multi-step forecasting is possible (contrary to ANN and kNN forecasintg)
- It is a deterministic regression/forecasting approach (similarly to ANN)